Knowledge-based support for evaluation of computer-based products

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades einer
Doktorin der Wirtschaftswissenschaften (Dr. rer. pol.)

genehmigte Dissertation

von
M.Sc. Tatiana Deriyenko
geboren am 17.04.1989
in Leningrad

Eingereicht am: 21.02.2019
Disputation am: 09.04.2019
1. Referent: Prof. Dr. Dirk C. Mattfeld
2. Referent: Prof. Dr. Mark Vollrath

2019

Disclaimer

The results, opinions and conclusions expressed in this thesis are not necessarily those of Volkswagen AG.

Abstract

Modern computer-based products are actively finding new applications in our daily lives. Smartphones, vehicles and home appliances are getting "smarter" in their effort to provide more assisting and entertainment services to their owners. From the development perspective this often requires significant financial investments and inevitably brings up a number of efficiency-related questions. For instance: is the new functionality well-accepted by the users or is it just overloading the product? Should all functions be further supported or have some of them become obsolete? Are there some goals that the users cannot achieve with the product? Is the overall product perception positive or is there a room for improvement?
In order to answer such questions and to tailor next product versions to what their users need, an extensive and comprehensive *evaluation of product quality* has to be performed. In case with complex devices this is a knowledge-intensive process that requires considerable human expertise and effort. Identification of relevant data sources, their processing and correct interpretation is challenging. For example, if a function is called less often than expected, does this indicate that it is non-compliant to the user needs? Which aspects may have caused this situation? How can user opinion or other sources be incorporated to reveal these causes?
We claim that information technologies are capable of helping developers to infer the required knowledge from various data on product usage. An introduction of a *knowledge-based system* can support the evaluation process and open up new possibilities for its efficiency increase. In this thesis we present a *knowledge model* to establish a basis for such systems. The model covers the user and the developer views on the products being in operational use. This includes generic knowledge on the user and the product, as well as knowledge that can be acquired during their interaction and can be used to improve the product.
The constructed model is well-suited for building a knowledge-based system upon it. In order to illustrate this, we create a system *prototype* using *ontological* technologies and demonstrate it on an *in-vehicle infotainment system*. The prototype receives user feedback and data on the product usage and suggests aspects that might be deviating from the user needs. The thesis is wrapped up with a conclusion and an outlook into future research.

Zusammenfassung

Moderne computerbasierte Produkte erobern immer mehr Anwendungsbereiche in unserem alltäglichen Leben. Smartphones, Fahrzeuge und Haushaltsgeräte finden zunehmend intelligente Ansätze, ihren Nutzern Hilfs- und Unterhaltungsfunktionen anzubieten. Aus der Perspektive der Produktentwicklung erfordert dies oft erhöhte finanzielle Investitionen und wirft damit unweigerlich eine Reihe an effizienzbezogenen Fragen auf, wie zum Beispiel: Wird die neue Funktionalität eine hohe Akzeptanz bei den Anwendern genießen oder wirkt das Produkt dadurch überladen und unübersichtlich? Sollten alle Funktionalitäten weiter unterstützt werden oder sind einige mittlerweile obsolet geworden? Kann der Anwender das Produkt zu allen von ihm gewünschten Einsatzzwecken verwenden? Ist die Wahrnehmung des Produkts insgesamt positiv oder gibt es Potenzial für Verbesserungen?

Um solche Fragen zu beantworten, und um folgende Versionen eines Produktes auf die spezifischen Bedürfnisse der Nutzer hin anzupassen, müssen umfangreiche Bewertungen der Produktqualität durchgeführt werden. Handelt es sich bei den Produkten um komplexe Geräte, so bedingt dies wissensintensive Prozesse, die typischerweise ein hohes Maß an Fachexpertise benötigen und erhebliche Arbeitsaufwände erfordern. Die Identifikation relevanter Datenquellen, sowie die Verarbeitung und Interpretation dieser, stellen eine Herausforderung dar. Wenn beispielsweise eine spezifische Funktion deutlich seltener verwendet wird als erwartet, bedeutet dies, dass es die Erwartungen der Nutzer nicht erfüllt? Welche Umstände könnten diese Situation hervorgerufen haben? Wie können Meinungen von Anwendern oder andere Quellen in den Produktqualitätsbewertungsprozess eingebunden werden, um Ursachen ermitteln zu können?

Diese Arbeit verfolgt die These, dass Informationstechnologie Entwickler dabei unterstützen kann, die benötigen Einsichten aus Produktnutzungsdaten abzuleiten. Die Einführung eines wissensbasierten Systems kann den Produktqualitätsbewertungsprozess unterstützen und eröffnet neue Möglichkeiten der Effizienzverbesserung. Diese Arbeit stellt ein Wissensmodell vor, welches die Basis für solche Systeme sein kann. Dieses Modell berücksichtigt die Perspektiven der Entwickler und der Anwender der im Einsatz befindlichen Produkte. Dies beinhaltet sowohl generisches

Wissen über Anwender und Produkt, als auch Wissen, das während der Interaktion von Anwender und Produkt erlangt und zur Verbesserung des Produkts verwendet werden kann.
Das entwickelte Modell ist geeignet, um darauf aufbauend ein wissensbasiertes System zu implementieren. Um dies aufzuzeigen, wird ein solches System mit Hilfe von ontologischen Technologien prototypisch entwickelt und am Beispiel eines In-Fahrzeug Infotainment Systems demonstriert. Der Prototyp empfängt Nutzerfeedback und Daten zur Produktnutzung, und generiert Hinweise, welche Bereiche den Erwartungen der Nutzer möglicherweise nicht entsprechen. Die Arbeit wird abgeschlossen mit einer Konklusion und einem Ausblick in die zukünftige Forschung.

Contents

Chapter 1

Introduction

1.1 Challenges of product evaluation

Various computer-based products have become an integral part of our everyday life and increasingly gain in importance. A survey for 2019 forecasts a more than 50% growth of smartphone users worldwide in comparison to 2014 [1]. For 2018 the market share of electronic books was estimated to exceed one fourth of total books sales [2]. In the automotive market the number of sold vehicles supporting infotainment services, such as online news reports, social networking or music streaming, expects a 7.5 time growth compared to 2012 [3].
Along with expanding the target audience, modern products are rapidly developing in terms of functionality. In the sequel we use the infotainment systems as an illustrative example. Early in-vehicle systems were relatively simple and had a purpose of providing the driver with such crucial information as speed or gas level. Later on, radio and some other basic entertainment functions were added. Modern infotainment versions are covering a much wider range of applications. In particular, the systems are able to deliver traffic and navigation information or information on the vehicle status [4]. The recent developments include smartphone functions, intelligent voice-operated assistance, gesture control and many others. Such extended features provide benefits for their users. However, at the same time, they also result in products becoming more complex, harder to develop, to maintain and to operate. The advanced multi-modal interfaces and a wide variety

of configurations increase the testing complexity [4]. This can be reflected in a significant number of problems encountered by the users: a recent survey by J.D. Power shows that infotainment-related issues account for 20 % of all problems reported by car owners and represent the most problematic area for the majority of vehicles [5].

The occurring issues result from different characteristics of the products. A particular user need, for instance, can be unsatisfied due to missing or inoperable functions, as well as due to functions "hidden" within a complex menu structure and being inaccessible or unclear. When all functional needs are met, the product can still be inconvenient to use, slow in operation or visually not attractive. Some of the existing functions may not be needed or even distract the user from performing his main tasks [6]. We distinguish the following major groups of factors leading to the aforedescribed cases:

different views — The users and the developers speak "different languages" [7]: the two parties normally have different backgrounds and, as a result, divergent views of the needs and of the product.

operational use — It is challenging to cover all user types within the development, especially for off-the-shelf products [8]. As a result, some specific requirements can be missed. Moreover, the actual usage context can as well vary from the one anticipated by the developer [9]. Finally, the user needs keep evolving over time, in particular, during the product operational use [10].

Fulfilling or even exceeding customer expectations can be considered as a major success of product development. However, the design of an "ideal" product is hardly attainable due to the described reasons. This shifts the focus from the initial development to the improvement of already existing products. By timely and competently adjusting the design, increasing performance or adding new features, the developers can achieve a higher user satisfaction.

A key step to planning corrective actions is evaluating products by checking their *quality* [11]. The results of the evaluation have to support the developers in making decisions on product improvement. In this respect, it becomes particularly

important to thoughtfully address the aforedefined factors within this process.
The *different views* can massively complicate the evaluation. For example, the developer may assume that the product is fully compliant to the user needs when all required functionality is implemented and no technical problems are detected. At the same time, the user can experience difficulties with finding certain settings and be highly dissatisfied. According to his or her viewpoint, however, these settings would rather be classified as missing than as "hidden". Tailoring products to the actual user needs therefore turns into a challenging task.
The second group of factors shows that this goal can also become a "moving target": the actual situation during the product *operational use* can not only deviate from the one foreseen at the development stage, but also keep further varying depending on the context. For example, gaining experience with other products can slowly shift user preferences. A quick change can be caused by the user fatigue or other health conditions. This imposes a major challenge, since verifying product compliance in a testing environment does not necessary assure user satisfaction during the actual usage. As a remedy, the dynamic nature of quality can be addressed by continuously checking products-in-use against current user needs. It implies that the product is evaluated iteratively and the results can be used to plan corresponding improvement changes [12].
In this thesis we focus on checking the *quality of products being used* by various customers. This approach can nowadays be facilitated with an increased availability of data on product usage. Along with traditional ways of collecting user feedback, like questionnaires or complaint services, new sources emerge, such as social networks, Internet blogs and discussion boards. Vehicles, smartphones and other modern products become a source of data themselves [13] by recording occurring events and thus expanding opportunities for the usage observation. Yet the analysis of these sources can be complicated due to their large volumes, diversity and low interpretability. The latter two aspects can be summarized as one major factor challenging the evaluation process:

data heterogeneity The data sources referring to the product usage can have a heterogeneous nature, by reflecting various aspects of the user-product interaction and by providing *different views* on them.

The heterogeneity of the sources becomes especially apparent if we explore available data on the usage of a particular infotainment system. As the first source, we consider various comments left by the users in the Internet. Such messages, posted in social networks, discussion boards and blogs, express user view on the product. This includes positive feedback, complaints, questions, requests for help, suggestions for improvement and other types of messages. How can this subjective input be used to find out which product characteristic should be adjusted within the production?

As another source, we pick up vehicle logfiles showing how exactly the infotainment system is used. Such sources are normally designed to track technical errors and mostly reflect the development perspective. A typical structure captures occurred events and provides corresponding timestamps (for example, "12:01 ButtonX State: Pressed"). How can this source be used to define if the users are satisfied with the product usage?

The interpretability of such heterogeneous sources on product usage is to a large extend determined by the views that they reflect and the view that is taken on them. However, in particular cases they can not only reveal important insights, but also mutually empower each other. While one of the sources can signalize that something is going wrong, another can shed the light on the reason for that. An illustrative example from the automotive field is a 3-blink turn signal used for lane changing. A conducted analysis of driver's behaviour has shown that this feature was used less often than expected. The reasons for this deviation were not clear: was such turn signal not needed? Were the drivers not aware of it? By analysing the customer feedback from social media it was determined that the function was needed by some of the drivers, but the switch positioning was unsatisfying[1].

The above example illustrates how consideration of different data sources can support informed decision-making on product improvement. However, which information can and need to be extracted for that? We consider these questions in more detail in the next paragraph.

[1] Cavaretta, M. (February 2014). The Structure Show [Audio clip]. Retrieved from: https://soundcloud.com/gigaom-structure/fords-data-scientist-keep-all/s-iV4AH

1.2 Research questions

Efficient knowledge management can play a key role in the development [14]. The evaluation of complex computer-based products is a knowledge-intensive task requiring considerable human expertise and effort. The defined challenges with *different views* and the subsequent *heterogeneity* of the data sources reveal the need for a comprehensive approach to structuring and utilizing relevant information to support this process. Our intention is to help developers to acquire the required knowledge from heterogeneous sources referring to the product usage. This implies introducing a source-independent solution applicable to different products and covering the two described viewpoints on them.

Various aspects of structuring knowledge for product development have been addressed in the literature, in particular, by capturing some information on the product [15],[16],[17], on the engineering process [18],[19] or on the customer requirements and preferences [20],[21],[22]. In the field of quality conceptualization multiple models [23],[15],[24] have been developed to provide a shared understanding of the main concepts. Several solutions have been suggested to support multi-perspective evaluation of services, in particular, of electronic government [25],[26]. In the discipline of human-machine interaction different models of user behavior were developed to understand user experience with the products [27],[28]. Regardless of the focus, these solutions do not explicitly aim at the evaluation task.

We consider models for quality evaluation support in detail. The review conducted in accordance with the guidelines for systematic reviews [29] is presented in chapter 5. While some researchers choose user feedback as an input for detecting problematic product aspects [30],[31], others focus on its technical performance [32],[33]. Yet the review determined no comprehensive solutions for the evaluation of products being in operational use. The identified approaches have a focus on either the developer or the user viewpoint, thus missing one of the perspectives. We aim at closing the identified gap with the goal formulated as follows:

Support quality evaluation of computer-based products-in-use
by designing a model
to facilitate structuring and utilization of relevant information
covering both the developer and the user views

The four research questions following from this goal definition are presented in Figure 1 and are described below:

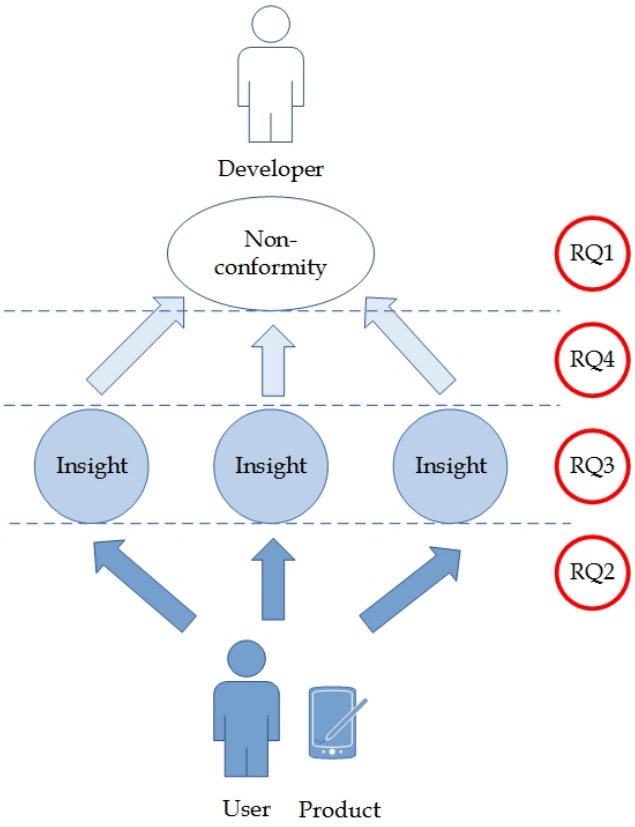

Figure 1: Overview of the research questions

RQ1 How can *product non-conformities* to the *actual user* needs be defined? The objective of supporting quality evaluation requires revealing product aspects that do not conform with the actual user needs. With the first research question we aim at formalizing the target output for the evaluation.

RQ2 Which *aspects of product usage* need to be considered within the evaluation? The user needs are intangible and can be even be not stated by the users, as the users themselves are not always aware of what they need [8]. With the second research question we focus on determining the input required to define product aspects that do not conform to the user needs.

RQ3 Which *relevant information* can be derived from analyzing product usage? With the third question we aim at defining which insights received from the input can be utilized for product evaluation. For instance, if analysing product usage shows that some feature is not used as often as expected, can this be considered as a useful insight for quality improvement?

RQ4 How can *meaningful conclusions* be derived from the acquired information? The purpose of the last question is defining how the revealed insights can be interpreted in order to be utilized for further product development. This point includes addressing the different viewpoints.

In the next paragraph we outline the approach we are following to answer the research questions and to create the required solution.

1.3 Knowledge modeling approach

In order to design the model for product evaluation support, we consider the field of KBS (knowledge- or knowledge-based systems, also named "expert systems" [34]). The main idea of KBS applications consists in providing required knowledge to an information system and making it act similarly to a human expert in solving a particular problem. The introduction of such system can facilitate the evaluation as follows: upon receiving the domain knowledge and the data on product usage, a KBS can suggest product aspects that do not fit the current user needs. The resultant output would then be interpreted by experts and used to adjust the product correspondingly.
Within this thesis we build a foundation for such KBS by designing its core component, a computer-interpretable *knowledge model*. This implies defining and representing the required relevant knowledge in a concise and unambiguous formal way. The resultant model is applicable to various computer-based products and is suitable for constructing a KBS over it.
In order to create a holistic model we follow *design science guidelines* for information systems [35] and specify them in accordance with the CommonKADS *knowledge-based system design methodology* [34]. The Figure 2 illustrates the structure of the thesis with a business process model[2]. The steps of the process are

depicted with rectangles, the annotations show corresponding chapters of the thesis. The solid line stands for the sequence flow, the dotted line together with the labels shows the main output of each step.

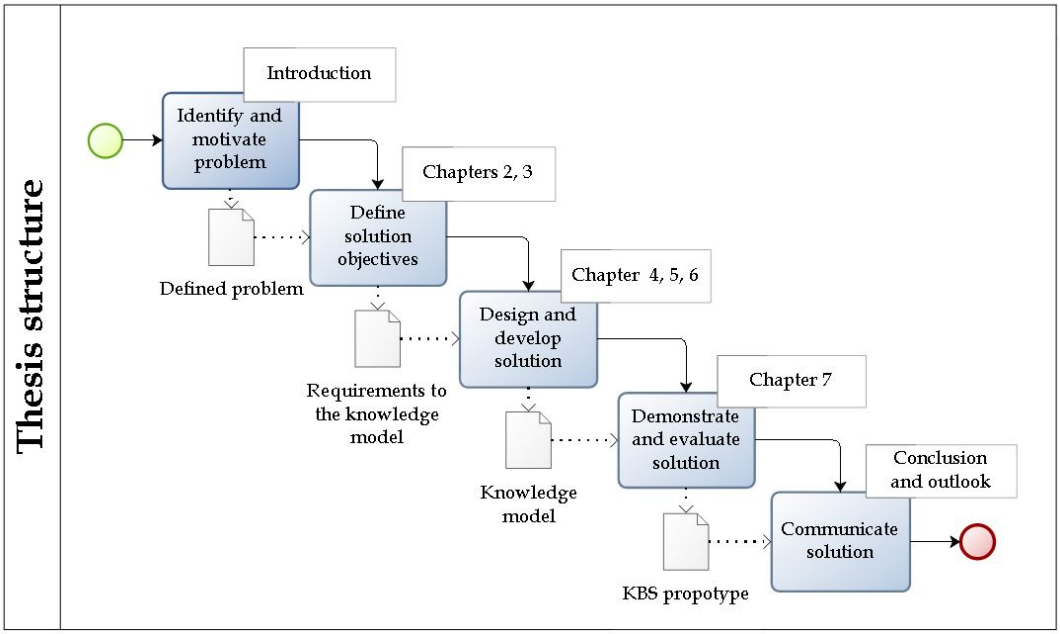

Figure 2: Overview of the thesis structure

We have defined the importance of the problem and outlined the potential solution in terms of knowledge modeling. The further steps are listed below:

1. *Definition of solution objectives.* The requirements to the target solution are derived from a domain context exploration. In chapter 2 we analyse the main characteristics of human-computer interaction. In chapter 3 we provide a background on product quality evaluation. As an outcome, we formalize a set of requirements to the designed model.

2. *Design and development.* In chapter 4 we explore foundations of knowledge modeling and of KBS. The chapter 5 presents the results of the state-of-the-art review demonstrating that none of the existing artifacts solve the defined problem. In chapter 6 we propose our solution by describing an implementation-independent *knowledge model.*

[2]Business Process Model and Notation: http://www.bpmn.org/

3. *Demonstration and evaluation.* In chapter 7 we build a prototype KBS based on the outcome of the previous chapter. The step corresponds to *design model* generation. The application of the prototype is illustrated in a case study with an infotainment system to demonstrate that it can serve the required purpose.

4. *Communication.* With the last chapter we wrap up the thesis by emphasizing the importance of the problem, as well as the novelty and the effectiveness of the proposed solution. The answers to the research questions are summarized. An outlook to the future research is provided.

With the next chapter we start formulating the requirements to the knowledge model by investigating how users and products interact with each other.

Chapter 2

Human-machine interaction

Abstract. *In order to understand the nature of product issues that have to be revealed within the evaluation process, we consider how and why the products are commonly used. The following chapter investigates the main characteristics of this process by considering both the human and the product perspectives. As the user-product system is open, we take into account the context in which the interaction occurs. In order to ensure the model applicability to various domains we follow a high level of abstraction. The review is concluded with a set of derived requirements to the knowledge model.*

2.1 Human perspective

As a starting point for building the knowledge model for product evaluation support we analyse the user perspective and explore the main human behavior principles with a particular focus on the interaction with products. Additionally we provide an overview of various contextual factors that can influence the behavior.

2.1.1 Human as information-processing system

Human mind in a simplified form can be considered as an information-processing system that consists of three interacting subsystems [36]. These subsystems are briefly described below:

perceptual The perceptual subsystem transforms different sensations detected by the human body into internal representations of his or her mind.

cognitive The cognition connects the perceptual and the motor subsystems by guiding human behavior.

motor The motor subsystem translates human thoughts into corresponding actions of the body.

The perceptual and the motor subsystems communicate with the surrounding world via multiple channels. This makes this communication *multimodal*: human normally combine spoken language with gestures, mimics and non-linguistic sounds, such as laughs or coughs. The utilized channels are not independent on each other: the communication via one of the modalities depends on the information simultaneously communicated via other modalities [37]. A simple example is a combination of voice and gestures in a conversation. The same holds for interacting with products: human can involve the whole spectrum of communication options when they are supported by the products.
Regardless of the product or the modality used, the cognition guides human interaction according to some goal. As defined by Rasmussen [38]:

> Humans are not deterministic input-output devices,
> but *goal-oriented* creatures, who actively select their goals
> and seek the relevant information

How do human behave in order to achieve their goals? Further behavioural principles and features are considered in the next section.

2.1.2 Human behavior principles

The prominent models of human behavior commonly represent it as a multi-level concept. This concept can therefore be considered hierarchically, at different *levels of granularity*. For instance, according to Leontyev, the behavior is defined with high-level activities. These activities are realized through a set of goal-oriented actions. The goals can be decomposed into sub-goals and so forth. The actions are implemented through lower-level operations, which are typically not recognized by the human [39].
Rasmussen similarly classifies human performance into three levels. The behavior

at the top level is following an explicit goal according to a prepared plan. The next level represents the behavior in a familiar situation, which can be controlled by a stored rule, but often has no defined goal. Finally, the last category defines the performance that follows some intention and has no conscious control [38].
The models presented by Card et al. as well propose multistep concepts. The behavior is considered as a sequence of operators representing elementary perceptual, motor or cognitive acts. In order to achieve their goals human use certain procedures named methods. These methods can be specified at different levels. If more than one method is available, the decision on the suitable one bases on a selection rule [40],[36].
The goals and the corresponding tasks that human follow can have a different priority and importance. For example, the primary tasks of a driver can be setting up a navigation goal and driving, while using air-conditioning or a telephone can play a secondary or even a tertiary role [41].
When trying to attain a goal, human follows a *rationality principle*, i.e. acts rationally [36]. That implies seeking the optimal path of least effort [42] while maximizing the benefits [43]. The path can be defined as a sequence of taken actions at different level of detail. The utilized resources, like the time spent on achieving the goal, should be minimized.
However, human abilities to make rational decisions can be restricted due to a lack of information, time constraints or cognitive limitations [44]. What is more, some actions can be defined as impulsive and have no particular purpose [43]. The behavior can thus be *error-prone* and not following the optimal path [45]. The errors can be defined as deviations from the user intention, expectation or desirability. That means that something was performed, but not intended by the actor [46]. Along with the above-mentioned factors, such deviations can be related to the process of human learning and adaptation or to intrinsic human variability, as the behavior evolves in time [42].

2.1.3 Context influence on human behavior

The users can differ in their knowledge on the product or on other systems, in their motor skills on input devices or in technical abilities [40]. These fac-

tors can have a significant influence on the product usage: according to Bevan, "a product which is unusable by inexperienced users may be quite usable by trained users" [47]. An illustrative example is the situation with early copiers that often caused confusion regarding handling procedures for the originals, supplies and copies. A deeper analysis has revealed that the users were used to traditional paper workflows. As a solution, these routines were adopted in the copier design to make it look like a desk with drawers to place paper in and to follow the understandable flow [48]. Modern copying devices do not require such features.

Along with the gained experience and skills, human performance depends on multiple other factors. This can be internal properties that are inherent for a person. The simple examples are the age group that the human belongs to, his or her physical capabilities or personality trait. The internal factors can have a temporal nature, for example, health conditions or fatigue.

At the same time, every living organism is an open system [49]. This means that it is also important to consider the external context, i.e. the world surrounding the human. This includes such physical factors as temperature or light, as well as the social or organizational context [50]. A simple example of the context influence in the everyday life is a situation when two people speak different languages and can communicate with each other only with mimics and gestures [37]. The cultural differences can appear, for instance, in different meaning of symbols, gestures or colours [33], as well as in different date format, units or reading direction [51]. The external context thus plays an significant role in human interaction with the products, as such factors can strongly influence the usage of the devices.

As a summary of this section, we conclude that human behaviour can be considered hierarchically, at different levels of granularity and can be represented as a sequence of actions guided by some goals. The human tend to act rationally in reaching these goals, however, their actual behavior can be not optimal. Finally, the behavior depends on various human-related factors, e.g. on the physical capabilities or experience, as well as on various environmental factors.

2.2 Product perspective

In this section we examine the product perspective similarly to the previously considered human perspective. This includes exploration of product "subsystems" and performance principles, as well as of contextual factors that can influence it.
The products that we focus on are complex devices designed to have multiple uses. For example, possible applications of the infotainment systems include navigating to a geographical goal, making phone calls or playing music. Such usage scenarios are implemented through a set of different functions provided by the product. Every function defines what a product entity is supposed to do [52].
The functions are accessible for the users via the product interface. The input-output devices utilized for that can include keyboards, mouse, trackball, touchpad, touchscreen, stylus, eyegaze, displays, joystick and others [33]. Modern computer-based products are commonly multimodal, supporting different communication modes at the same time, e.g. speech or gestures [53]. The set of functions provided by the products can therefore be used via the interface in different modalities, either simultaneously or subsequently [54],[55]. Depending on the configuration, the products can offer different sets of functions and modalities.
The products are designed by the product developers to fit user needs. The planned purpose of the interaction with the product is therefore to accomplish some of the human goals and to satisfy the users. The functions can constitute a hierarchy [56]: the product functionality can be considered at different granularity levels in correspondence to the user goals. While some of the functions directly correspond to these goals, others play a secondary role in their achievement. Two illustrative examples from the in-car infotainment field are setting up a navigation goal or updating the maps to keep them up to date. Some product functions can be executed in background and even be invisible for the user [28].
The product performance depends on the internal context, for example, on its technical conditions. At the same time, the performance can also be influenced by various environmental factors, like the light or temperature [50],[57]. An example for the infotainment system could be a poor radio reception in tunnels or a delayed display response during cold weather.

2.3 Requirements to interaction modeling

In our research we are considering the system of one user and a product interacting with each other. In the previous sections we have explored the perspectives of each of these actors. As the next step we formulate the interaction characteristics to be addressed in the knowledge model design. These characteristics are listed below:

REQ1 action — The interaction is represented by a flow of actions taken by the user and the product. Each of these actions involves product *functions* using the interface in some *modality*.

REQ2 granularity — The user and the product behavior, as well as the user-product interaction can be considered hierarchically at different levels of *granularity*.

REQ3 context — The *context* influencing the interaction can be related to the user factors, to the product conditions and to the environment, i.e. to external factors.

REQ4 properties — The interaction depends on product configurations, for instance, on the availability of functions in different modalities. These properties can be considered as a part of the context.

REQ5 user fit — The interaction with the product is user *goal-oriented*. It should satisfy the user, the goals are to be achieved efficiently and effectively, i.e. with a minimal effort from the user side.

The first four outlined principles of the user-product interaction are important to understand when and where the product issues occur. Each of them therefore will be addressed when designing the knowledge model. What does the fit to the user needs mean? The latter principle refers to the nature of the issues and belongs to the field of product quality. We explore the user-product system from the quality perspective in the next chapter.

Chapter 3

Fundamentals of quality evaluation

Abstract. *This chapter introduces the fundamentals of quality evaluation. We start with outlining the role of the evaluation within product development and continue with providing an overview of existing views and models of quality, as well as of evaluation approaches. As a result of the review, we define a further set of requirements to the designed knowledge model.*

3.1 Views of quality and quality models

With the previous chapter we have concluded that product development should focus on the user needs. The major goal is thus to achieve user satisfaction by providing a high-quality product. However, the definitions of quality may be vague and even controversial. In order to describe the relevant viewpoints for the model, we determine the role of quality, explore the common views of this concept and present the major quality models that reflect them.

3.1.1 Product evaluation and quality

A common approach to achieving a higher quality is implementation of a continuous improvement strategy. The strategy can be illustrated with a PDCA (Plan-Do-Check-Act) cycle inspired by Deming [58]. Below we briefly present a cycle modification by Ishikawa who defines it with the following four steps [12]:

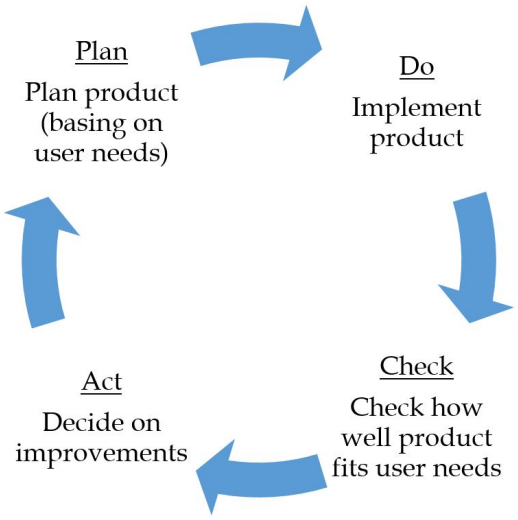

Figure 3: PDCA cycle for product design

1. *Plan*: determine the goals and a solution for reaching them.
2. *Do*: implement the solution, perform user education and training.
3. *Check*: evaluate the results of the implementation.
4. *Act*: take a needed action. If the results of the evaluation are unsatisfactory, go to the *plan* step and start the cycle again.

In terms of product development, the aforedescribed steps imply that the product is first planned and implemented in order to satisfy the determined user needs. With the next step the evaluation is performed to check if this goal is achieved. Basing on the received results, a plan for corresponding "improvement" changes is derived. These changes can refer to the product, for example, to a performance increase or to design modifications, as well as to the users, for example, to conducting trainings or providing extended manuals. A simplified version of the cycle is presented in Figure 3.
The PDCA cycle can be repeatedly executed at different development stages: an intermediate product configuration can be checked against certain technical specifications, while a complete product can already be validated by the actual users.

In any of these cases the central role is dedicated to the product *quality*: the goal of the evaluation is to assess the quality with regards to the specifications, to the user expectations or according to some other criteria. Depending on the development stage and on the viewpoint taken, the quality receives different definitions. In the next section we consider this aspect in more detail.

3.1.2 Views of quality

In this section we present an overview of the major views of quality suggested in the literature. Garvin proposed a common classification of the main viewpoints of product quality [59] that were further adapted by Kitchenham and Pfleeger for the software domain [60]. These viewpoints are defined as follows:

Product — The product view focuses on its internal, inherent characteristics. This view is context-independent and can be assessed objectively.

Manufacturing — The manufacturing or the developer view defines quality as conformance to some specifications or standards.

User — The user subjective and highly personalised view considers quality with regards to meeting user needs within some context.

Transcendental — The transcendental view defines quality as an ideal ethereal concept that can be recognized, but not measured.

Value — The value-based view considers quality with regards to the costs that the user is ready to pay for the product.

The focus of the development should shift from the *product* to the *manufacturing* view and then from the *manufacturing* to the *user* view, as the product moves from the design stage to the market [59].
Another common classification approach makes a distinction between the *perceived* (or *subjective*) and the *objective* quality. Shewhart defines the objective quality as being independent on the existence of a human, while the subjective side is directly related to what a human wants [61]. According to Bevan, the user-perceived quality bases on inaccurate judgements [62] and therefore corresponds to the user view.

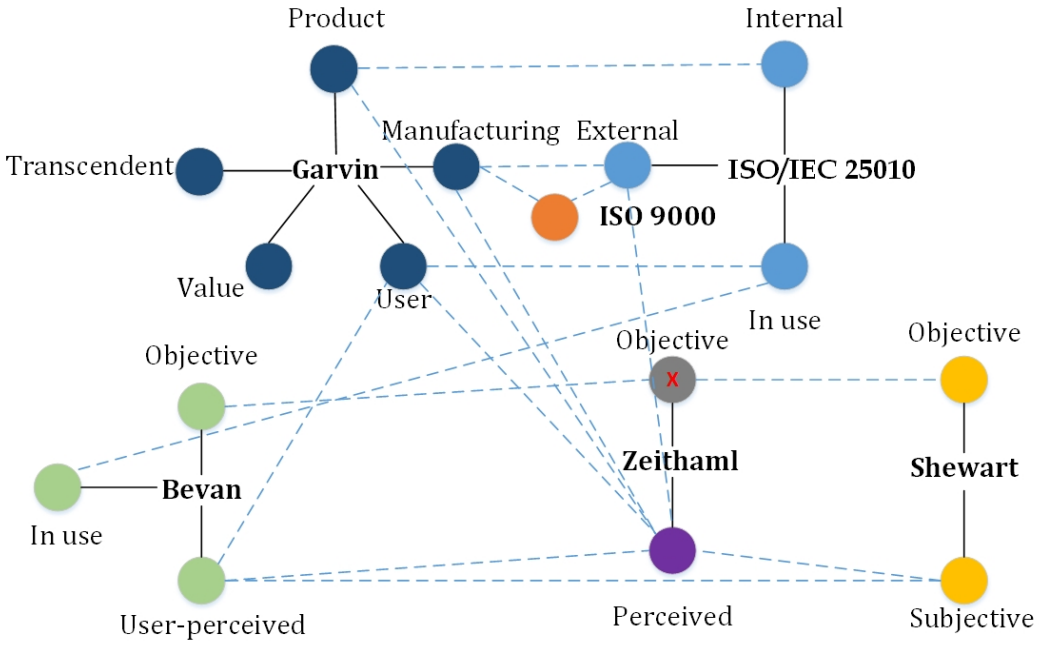

Figure 4: Overview of existing views of quality

However, according to Zeithaml, as the manufacturing and the product views base on some perceptions as well, the objective quality may be non-existent [63].
An overview of the major quality views as well as of the relations between them is presented in Figure 4. In the next section we consider the quality models reflecting some of these perspectives.

3.1.3 Common quality models

The selected viewpoint defines which model should be applied to assess the product quality [11]. The *product*, *manufacturer* and the *user* views are commonly reflected in various quality standards. The ISO 9000, for instance, defines the quality in relation to inherent product characteristics fulfilling some requirements [64], while the ISO 9241 outlines the importance of product evaluation from the user perspective [50].
We consider the ISO/IEC 9126 [65] and a more recent ISO/IEC 25010 [66] standards that introduce the following two quality models addressing different view-

Model	View	Stage	Focus
Product quality model	Product (internal quality)	Development	Product static properties
	Manufacturing (external quality)	Development or operation	Product dynamic properties
Quality in use model	User	Development or operation	Interaction outcome

Table 3.1: Summary of views on quality and quality models

points: the *product quality* model and the *quality in use* model. We prepare an overview of the models' properties and present them in the Table 3.1.
The *view* refers to the perspective that the model reflects. The *stage* defines if the product is evaluated during the development, before it starts being used by the customers, or already during the operational usage. The *focus* determines which aspect of the product is considered during the evaluation: this can be static properties, properties measured during the product operation or properties describing the interaction of the users with the products. For each of these cases the models provide a set of *characteristics* that can be used to evaluate the quality. We consider these characteristics in more detail in the following paragraphs.

Product quality model. The first considered model covers two categories of product quality: the *external* and the *internal* once. The internal quality is defined by static measures of the product. Such properties can refer, for example, to the source code quality. The external quality bases on the behaviour of the product in a specified environment. This can be, for instance, a technical testing or an operational environment.
The described categories can be seen as related to the *product* and the *manufacturing* views correspondingly. The characteristics that are proposed for the evaluation include such product aspects as reliability, compatibility with other systems, performance efficiency and others. An overview of the characteristics is presented in Figure 5. The hierarchical representation illustrates that the characteristics can be broken down to a lower level of granularity.

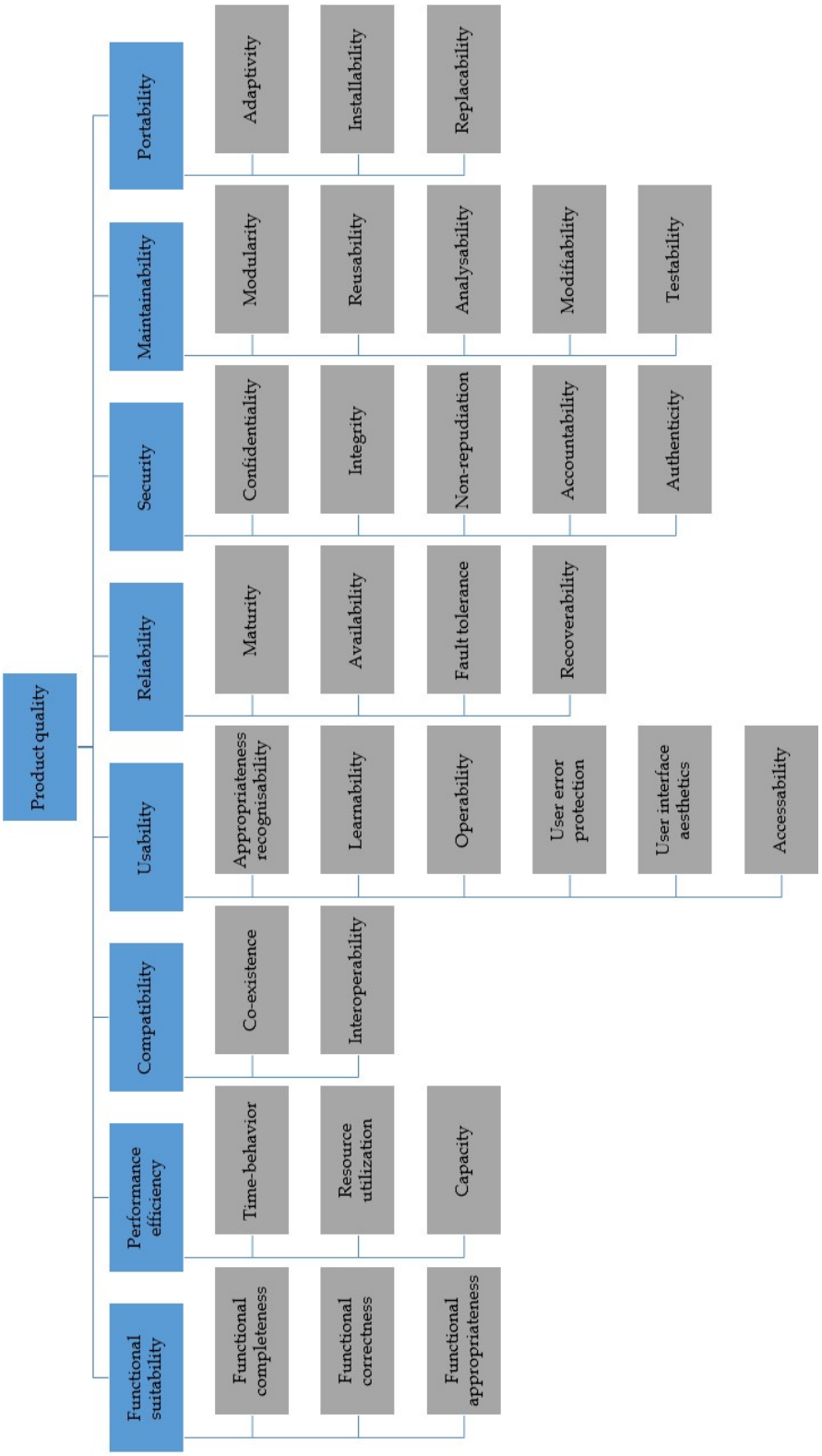

Figure 5: Characteristics from the product quality model (ISO/IEC 25010)

Quality in use model. The second model proposed by the standards defines quality characteristics with regards to the output of the user interaction with the product in an operational context. The quality thus represents a degree to which the product meets the user needs and provides the user view of it. This includes such aspects of the user-product system as efficiency and effectiveness in reaching some goals, user satisfaction covering such aspects like pleasure or trust, as well as freedom from risk.
An overview of the characteristics suggested by the quality in use model is presented in Figure 6. It can be observed that some of the characteristics can be decomposed into subcategories.

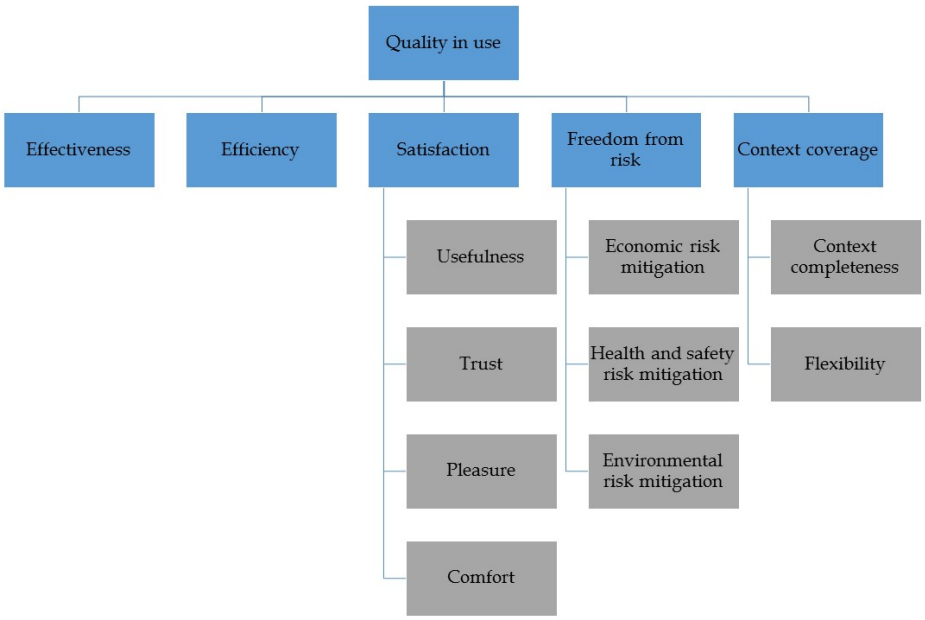

Figure 6: Characteristics from the quality in use model (ISO/IEC 25010)

Product evaluation approaches. We have outlined the quality models and specified that they can lay the focus on different aspects within the evaluation: on the static or dynamic properties of the products or on analysing the output of the user-product interaction. Below we provide some examples of the existing evaluation approaches in order to illustrate how the different focuses can be followed in practice.

Along with investigating the source code, the consideration of the static properties can include checking product interface accessibility [67]. Monitoring of the user-product interaction can be performed to check both the user and the machine performance [50]. In the product part, for instance, such parameters like the response time or uptime can be measured and compared with corresponding values from some requirements or standards. The analysis of the user performance can as well involve measuring the reaction time [40],[68]. The behavior rationality can be accessed [69], the errors made during the product usage can be detected [42]. The evaluation can also cover both the user and the product at the same time. For example, the time spent on completing some task using the product can be measured or the rationality of the user-product interaction can be explored [29].
The subjective view on the product and on the interaction with it can be fetched from the user feedback. Such input can reveal user motivation, feelings, beliefs and attitude. ISO 9241 defines the feedback during product usage as a base for system modernization [50]. The data can be collected by asking the users to "think aloud" during the product usage or by using various other sources such as complaint services or questionnaires. Social media is nowadays getting deeper integrated in our lives. People tend to share their concerns or recommendations with the Internet communities. As a result, different online discussions have proven to be a valuable source of information about product quality [70].
The common approaches for evaluating quality in use include analysis of the user-product interaction and of the user feedback [50], as both the objective and the subjective outcomes need to be considered [62].

3.2 Requirements to product evaluation

We have considered the views of quality and quality models and sketched some approaches and sources that can be used for the product evaluation. We now formulate the evaluation requirements to the designed knowledge model. As defined in the introduction, we aim at supporting the user and the developer views on the product. None of these views alone suffices for a comprehensive product evaluation due to the following reasons:

user view — The *user view* reflecting the user needs is subjective and may be inaccurate, as defined in the previous section. The users lack knowledge on the product structure and engineering.

developer view — The *developer view* does not directly reflect the user needs. What is more, it can as well be subjective, as it was defined in the previous section.

The objective actual user needs are not known and the knowledge derived from available sources can be subjective and incomplete. For instance, the user feedback can omit a direct reference to the product model, while the developer view can miss out the user cognitive side of product usage. As a conclusion, we formulate the following requirement:

REQ6 knowledge — The product evaluation has to operate under the condition of *limited knowledge* availability.

The final requirement we define follows from the consideration of quality models and is presented below:

REQ7 granularity — The product quality can be considered as a multi-level concept, at different levels of granularity.

In this part of the thesis we have performed an analysis of the domain context and defined a set of corresponding requirements to the model. In the following chapter we explore in detail how knowledge models and KBS are designed.

Chapter 4

Fundamentals of knowledge systems

Abstract. *The goal of the following chapter is to gain a comprehensive background on knowledge modeling and on KBS in order to pave the way for a solution conforming with the requirements defined in the previous chapters. We introduce the fundamentals of KBS, explore their design principles and take a close look at the existing ways of representing knowledge and building knowledge models. The chapter is divided into three corresponding sections, followed by a short conclusion.*

4.1 Introduction to knowledge systems

The concept "knowledge" has multiple, often contradicting interpretations. This section aims at determining the definition supported within the knowledge engineering community. After investigating the fundamentals, we describe the main components and principles of knowledge-based systems.

4.1.1 Definition of knowledge systems

In order to determine the notion of knowledge within KBS, we first briefly introduce a widely used Data, Information, Knowledge, Wisdom (DIKW) pyramid. The pyramid represents a set of interrelated concepts, which is also known as "knowledge hierarchy". This hierarchy originates from the work of Ackoff, who distinguished five concepts describing human mind content. The top concept is *wisdom*, followed by *understanding*, *knowledge*, *information* and *data* in a descending order. Every higher concept in the hierarchy includes all concepts lying below:

in particular, wisdom does not exist without understanding, the understanding is impossible without knowledge [71].
Further elaboration on the Ackoff's approach by Bellinger et al. excluded *understanding* from the hierarchy and defined it as a process supporting transitions from the data to wisdom [72]. The resultant hierarchy can be found in Figure 7.

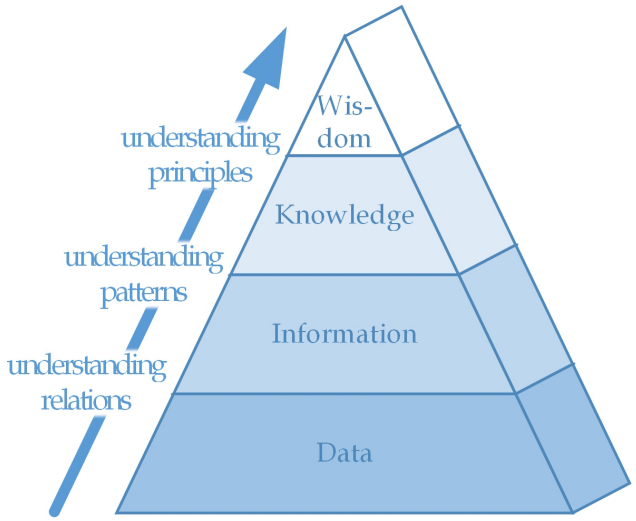

Figure 7: Transition from data to wisdom

Below we provide definitions for every components of the hierarchy and complement them with illustrative examples:

data — Data is a set of symbols that has no relations to other concepts, for example: "The user is not satisfied". The data can be usable or not.

information — Information is data that is given a meaning by establishing relational connections, e.g. between a cause and an effect. An example is: "The quality of the product is low, therefore the user is not satisfied". The meaning is not necessary useful.

knowledge — Knowledge is a collection of information with an intend to be useful. An example is: "When the product quality is low, the user can be not satisfied with it".

wisdom — Wisdom bases on an understanding of fundamental principles residing in the knowledge and rises some questions with no easily-achievable

answers. For example: "Achieving user satisfaction requires an absolute quality, which might not exist".

The *data*, *knowledge* and *information* are fundamental blocks of information systems. Yet the researchers follow different interpretations of these concepts [73]. We now determine the notion of knowledge that we follow.
The traditional transition from data to knowledge up the pyramid relies on manual analysis and interpretation [74]. The goal of KBS is to support this transition with information technologies. Such systems can be defined as *information systems utilizing expert knowledge to solve problems from particular domains*. In distinction to other systems, KBS utilize an explicit knowledge representation [34].
However, despite this definition, KBS rather deal with information than with knowledge. As stated by Wilson, "data and information may be managed, and information resources may be managed, but knowledge (i.e., what we know) can never be managed, except by the individual knower and, even then, only imperfectly" [75]. The knowledge therefore resides exceptionally in human mind. The purpose of the KBS is *aiding* a human in inferring useful knowledge from available information. The main expected benefits are increased speed, productivity and quality of human decision-making [34].
Within the knowledge modeling context we will imply *information* under the term "*knowledge*". The next sections explore the main principles of KBS: we consider a common framework for information systems design research and define how knowledge systems operate within it.

4.1.2 Knowledge systems within IS framework

In Figure 8 we present a framework suggested by Hevner et al. [76]. The framework defines the main concepts of the design-science research for information systems.

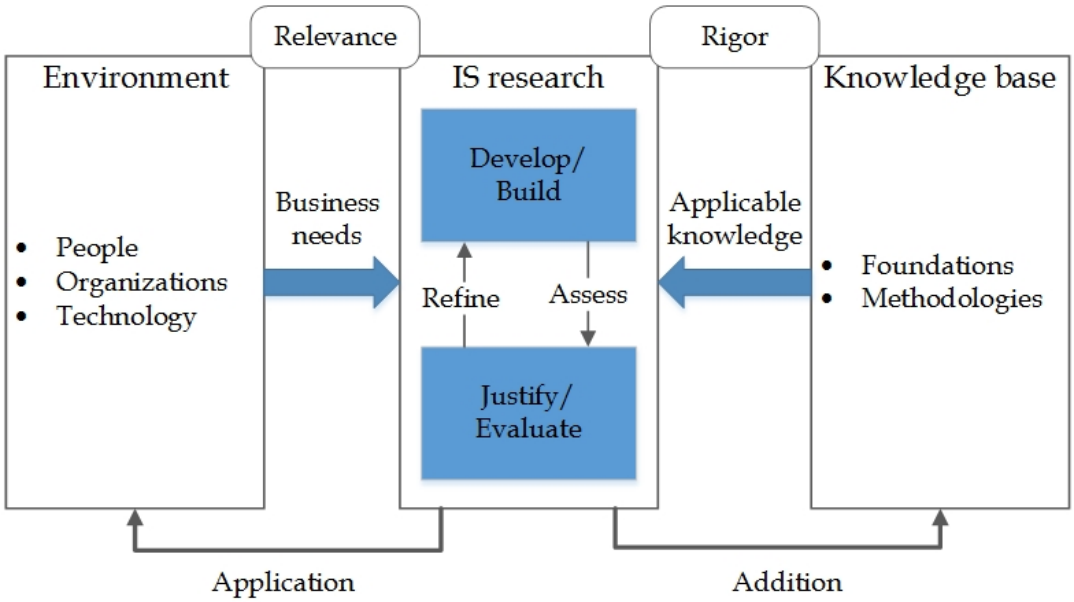

Figure 8: Framework of information system research

The *environment* block on the left side depicts people, organizations and used technologies that define the problem space. The *business needs* as perceived by people are determined by their goals, tasks, problems and opportunities. The *relevance* of these needs is evaluated within the context of organizational structure, business processes, infrastructure, etc.

The *information system research* depicted by the middle block addresses the defined *business needs* as perceived by a researcher. The corresponding activities include *developing* and *justifying* theories related to these needs, as well as *building* and *evaluating* some artifacts to meet them. The artifacts are rarely represented with fully-implemented systems, but rather with some contributions supporting their design and utilization. This includes *constructs* providing a language to define the problem and the solution; *models* representing the problem and its solution space; *methods* for a solution search; or *instantiations* demonstrating that these contributions can be implemented into a working system. The research *rigor* is achieved by using existing foundations depicted by the *knowledge base* block. The contributions are *added* to this base and *applied* in an appropriate environment.

The aforedescribed framework is fully applicable to the field of knowledge modelling, since KBS is a special case of information systems. The artifacts can thus

be represented by a full-scale or a prototype system, as well as by a novel knowledge model, language or method. In the next section we take a closer look at the components that constitute knowledge systems.

4.1.3 Principles of knowledge systems

The KBS operate two different types of knowledge: *factual* (or declarative) and *procedural* [77]. The factual type explains the nature of the things, e.g.: the customers use products. The procedural knowledge explains how these things work, e.g.: if a user is not satisfied, the product quality has to be improved.
The Figure 9 presents a UML component diagram to illustrate the KBS structure and to explain how the aforementioned knowledge types are applied within it. The rectangles represent the main system blocks. The arrows with a half circle at the end depict the interfaces that are required by the component, the arrows with a circle depict the interfaces that are provided by it.

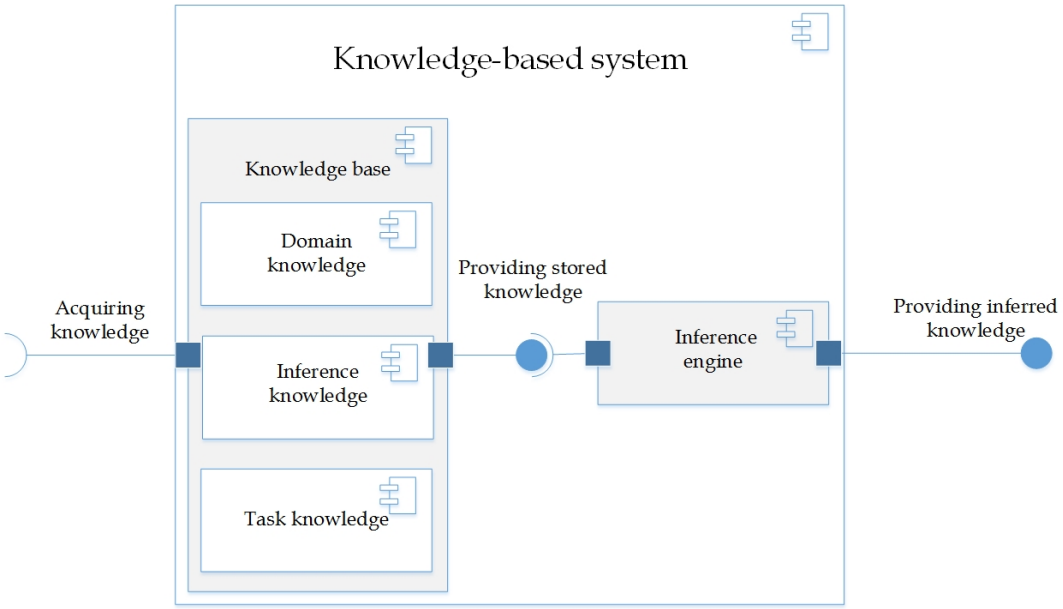

Figure 9: A common KBS system architecture

The central system component is a *knowledge base* that contains the following categories of specialized knowledge on the defined problem area [34]:

Domain — The domain knowledge expressing declarative facts by describing existing concepts and the relations between them.

Task — The task knowledge referring to the goal of the system and to strategies in achieving it.

Inference — The inference knowledge representing the procedural steps to be performed over the domain knowledge in order to achieve this goal.

The latter two points together are named *control* knowledge and are combined in problem solving methods [78].
The second component of the system is an *inference engine* that processes the facts stored in the base to infer knowledge required by the goal. This can be performed using the following two fundamental methodologies or their combination:

rule-based — The rule-based reasoning uses "if-then" rules to infer knowledge. This requires a set of pre-defined rules to be stored in the knowledge base. The reasoning process bases on deduction: if the "if" condition is true, the rule is applicable and the "then" conclusion is true as well. The search for applicable rules can be performed by backward chaining starting from a goal and looking for the rules containing it in the "then" part or by forward chaining starting with a known fact and searching for the rules that contain it in the "if" part.

case-based — The case-based reasoning uses knowledge on already solved problems from the past. New problems are approached by retrieving and utilizing solutions for the similar cases from the knowledge base. The retrieval is performed based on some similarity measure.

The last component, the *interfaces*, serves to acquire facts for the knowledge base and for delivering the inferred knowledge to the system users.

Below we present the three common steps that are followed to achieve that:

1. *Knowledge acquisition.* The goal of the process is to derive relevant knowledge from available sources, such as human experts or documentation, and to provide it to the system.

2. *Knowledge representation.* The acquired facts are represented in the system according to a knowledge model and are saved in the knowledge base.

3. *Knowledge inference.* Finally, the inference engine is applied to receive hidden knowledge from the facts and to provide it to the user. Additionally, the system can generate explanations for the inferences.

We have considered the main KBS components and their role in the system operation. In the next section we explore how knowledge-based solutions are designed.

4.2 Design of knowledge-based systems

We have introduced the fundamentals of information system research in general and of KBS specifically. In this section we take a closer look at the CommonKADs methodology that is used for knowledge solutions design [34]. The models suggested by the methodology are presented in Figure 10. We further consider some of them in more detail.

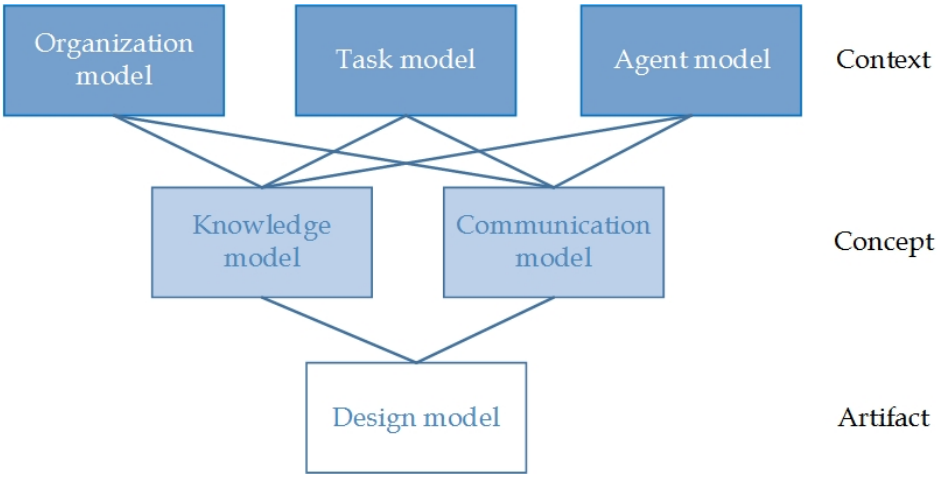

Figure 10: CommonKADs model suite

4.2.1 Task modelling

The organizational and the agent aspects of the defined problem were explored in the previous chapters. We therefore start the consideration with the task models. The tasks to be performed by the KBS are derived from the main goal that is followed. The systems can be applied for both the synthetic and the analytic tasks. In the analytical case some data about an existing system is utilized to characterize this system. The synthetic task receives some requirements to a yet non-existing system in order to generate its description. As we focus on already existing products, we further look into the analytical applications. These applications can be divided in accordance to the problem type as follows [34]:

classification The classification characterizes the system by assigning it to a class.

assessment The assessment characterizes the system in terms of a decision class.

monitoring The monitoring continuously analyses a dynamic system to conclude if its behaviour is normal or not.

diagnosis The diagnosis characterizes a malfunction of the system.

prediction The prediction constructs a system description for some timepoint in the future.

For each of the tasks the CommonKADs framework offers corresponding templates. Once the tasks and other requirements are defined, the development of the artifacts starts. We consider the corresponding procedures in the next paragraph.

4.2.2 Knowledge modelling

The communication model defines how the KBS has to interact with the world. The core design activity is the construction of a *knowledge model*. This process includes the following steps:

1. The *knowledge identification* includes specifying relevant sources, domain terms and existing model components.

2. The *knowledge specification* step aims at defining the target domain model for the KBS.

3. The *refinement* step is required to validate and tune the designed model.

The knowledge can be represented in different ways. However, despite a great variety of available formalisms, the resultant models commonly have the following two elements [77]:

concepts The concepts that are the basic units constituting knowledge.

relations The relations that link these concepts to each other.

The knowledge models are implementation-independent. In the next section we explore how they can be represented in design models.

4.2.3 Design model creation

The construction of an implementation-specific *design model* bases on the previously described modelling activities and includes choosing an overall approach to the design process, selecting an optimal knowledge representation and deciding on how to implement it with a particular software [34].
The effectiveness and efficiency of the KBS highly depend on the selected knowledge representation *formalism*. The formalisms differ in the degree of expressiveness, spanning from simple controlled vocabularies to extensive multilayered networks with a complex system of predefined relations among the knowledge concepts. Below we briefly explore the most prominent approaches.
Semantic network was first introduced by Masterman in 1961 [79]. The formalism represents knowledge as a system of nodes to depict the concepts and of directed arcs to define the relations between them. The two commonly used types of relations are "is-a", which stands for "is an instance of", and "a-kind-of" [80]. The structure of the networks supports determining how two concepts are related to each other, calculating their similarity, as well as inheritance reasoning assuming that all subconcepts in the hierarchy inherit all properties of their parental concept [77]. The formalism represents a simple structure supporting only binary relations and providing limited reasoning capabilities.

Frame is a representation formalism basing on semantic networks. The notion of frames was introduced in 1974 by Minsky as "data-structure for representing a stereotyped situation" [81]. Each frame has a number of slots with restricted or arbitrary values. The slots can be used to establish relations, e.g. if a frame x has a relationship to y, then one slot of x has a value of y. The formalism provides efficient mechanisms for inheritance reasoning. The advantage over semantic networks is the ability to define some procedures, thus combining factual and procedural knowledge. A procedure, for example, can be triggered when a new value is assigned to a slot. Early knowledge systems were widely utilizing a frame-based domain with simple *production rules* representing inference knowledge [82]. However, the modelling power of frames is relatively limited. In particular, it is difficult to represent negations and non-taxonomic knowledge.

Description logic as a knowledge representation technique stems from formalizing semantic networks and frames. The main difference to these formalisms is utilization of formal logic-based semantics. The logic describes the domain in terms of concepts, roles and individuals. A knowledge base consists of a terminological and an assertional parts. The terminological part describes properties of domain concepts, roles and relationships between them. The assertional part describes concrete situations by stating properties of individuals [83].

Ontology is the last domain knowledge representation formalism that we consider. The term has various definitions. We follow the one provided by Studer et al.: "Ontology is a formal, explicit specification of a shared conceptualization" [84]. This includes a hierarchical set of concepts with specific instances at the lowest level. The arbitrary or restricted properties describe various features of the concepts [85]. Effective mechanisms to maintain hierarchical and other dependencies among concepts are provided [86]. Additionally, the ontologies are well-suited for enhanced reasoning [87] along with the description logic inference. In the part of the inference knowledge there are multiple languages designed specifically for ontologies [88]. The *Semantic Web rules* (SWRL) [89] is one of the common examples. A more detailed description of the ontological formalism and of the corresponding design process principles can be found in the Appendix.

4.3 Summary of the chapter

In this chapter we have considered the foundations of knowledge models and systems and presented a corresponding design framework. We have also provided an overview of existing knowledge representation techniques and outlined that ontologies represent one of the richest formalisms that can be combined with enhanced inference approaches. We now use the fundamentals presented in this chapter to explore the existing KBS applications for quality evaluation. In the next section we consider state of the art in this field.

Chapter 5

Knowledge models for quality evaluation

Abstract. *Within this chapter we closely investigate the intersection of the following two fields: of product evaluation and of knowledge modeling. This is performed by systematically exploring scientific publications belonging to both of these disciplines. The objective of the review is to determine relevant artifacts, like knowledge models, prototype or fully-implemented knowledge systems, designed to support quality evaluation. This includes identification and evaluation of existing artifacts.*

5.1 State-of-the-art review process

In the introduction to this thesis we have provided a brief overview of related applications in knowledge modelling, including approaches for structured product description, human-machine interaction exploration, evaluation of service quality and others. Within this chapter we present a state-of-the-art review focussing specifically on knowledge modeling for *product evaluation support.*
The objective of the conducted review was to identify and to explore relevant knowledge artifacts designed for this purpose. These artifacts could be represented by knowledge models, prototype or fully-implemented KBS and other contributions to the field. The two reasons for performing the review are described below:

- to determine if any of already existing knowledge artifacts presented in the literature solve the defined problem.
- to acquire a rigorous background in order to properly position our research with regards to the existing knowledge-based solutions.

In order to achieve that we follow the guidelines for systematic literature survey from software engineering community [29]. The review was conducted in August 2018. We utilized Google Scholar engine[1] to systematically search for the research publications belonging both to the field of *knowledge modeling* and of *quality evaluation* of computer-based products. Various combinations of the keywords from each of the following categories were used for the queries:

- knowledge management, knowledge engineering, knowledge-based support, knowledge-based system, knowledge system, expert system, knowledge representation, knowledge model, information model, ontology.
- product evaluation, software evaluation, product quality, software quality, quality evaluation, quality assessment, quality measurement, usability evaluation, human-machine interaction, human-computer interaction.

Out of the identified publications we select those ones that refer to computer-based products and their quality. In order to structure and further review the remaining results we utilize the following set of questions:

state — Product in which state is considered? The stages when the evaluation can be performed are described in section 3.1.1.

view — Which view of the product is taken for the evaluation? Existing viewpoints are explored in section 3.1.2.

artifact — What kind of artifact is presented? The list of possible artifact types is presented in section 4.1.2.

task — Which task is followed by the developers of the artifact? The list of tasks is presented in section 4.2.1.

[1] Google Scholar: https://scholar.google.com/

knowledge What kind of knowledge is utilized? The types of knowledge are described in section 4.1.3.

As the next step, we evaluate the relevance of the identified artifacts by checking their applicability to the defined problem. This involves answering the first two questions. The following paragraphs present the results of the review.

5.2 Knowledge applications for quality evaluation

We group the detected artifacts according to the application fields and the tasks that they are designed for.

5.2.1 Fault detection and diagnosis

The combination of monitoring and diagnosis tasks finds various applications in the field of quality evaluation. Below we present some of the identified knowledge-based artifacts that fall under this category.

Technical fault detection and diagnosis. The first considered research field is technical fault detection and diagnosis (FDD). The *fault* is defined as an unpermitted deviation of some system feature from an acceptable condition. The goal of the monitoring is to detect such deviations and to determine as many details about them as possible. The details can in particular be represented by the fault type, size or location [90].
The diagnosis task aims at finding the causes of the detected faults. For that the observed effects are considered as *symptoms* pointing to these faults. The symptoms can be analytical or heuristic and be presented in the form of numbers or described linguistically by a human [90]. The input for the diagnosis is represented as a set of such symptoms and some knowledge on the analyzed process [32].
One of the earliest examples of knowledge systems for FDD is a DART system. The system was designed in a form of an automated consultant to support service personnel in the diagnosis of faults occurring in teleprocessing systems. The goal is to suggest the components that could be responsible for the observed faults and explaining these suggestions. As an input, the system uses human description of

the experienced problems. The set of rules utilized to find the faulty components was derived from the documentation and by interviewing experts [91].
The CAFD system for industrial robotics combines the analytical mathematics-based symptom generation with their heuristic evaluation based on available fault statistics, process history and human expert knowledge [92]. The COFES is an expert system focussing on monitoring and diagnosis of power plants operation basing on the knowledge on the power network structure and on a set of diagnostic rules derived from the past experience [93]. A diagnosis ontology for the automotive domain aims at describing knowledge that can be retrieved from available textual repair data [94].

Development process monitoring. The similar principles are applied in the ARROWSMITH-P prototype expert system. The system aids in detection and assessment of problems occurring during software coding and testing stages. The development process is checked against normal patterns derived from the past projects. If some deviations are detected, the system attempts to determine the reasons for them by using expert-defined rules. The reasons can include low productivity, unstable specifications and others [95].

5.2.2 Product selection support

The applications in software selection aim at supporting the process of choosing a proper product out of several options. A knowledge-based approach (KBS) uses knowledge on organization priorities and process structure to recommend a suitable IT application [96]. The ESSE system suggests software characteristics, metrics and methods for specific software selection problems basing on the experience from the past [97]. In the HKBS the rules represent knowledge about software evaluation criteria to capture user requirements, while the case-based reasoning is used to compare existing software packages with these requirements [98]. A concept for the selection support of geographic information systems is presented in [99]. In the aforementioned approaches the user needs are received as an input.

5.2.3 User feedback analysis

The last considered group of knowledge solutions aims at analysing subjective user feedback and making conclusions about the extent to which different product aspects fit the user needs. CASI expert system [100] and its modification [30] follow a rule-based approach for the evaluation of software usability by analysing subjective answers of the evaluators. Akinnuwesi et al. presented a NFES expert system that uses end user ratings as an input for the evaluation of distributed systems. The knowledge base includes the requirements of the organization and software constructs [31]. Leoprairote et al. suggested using a software quality ontology to classify product reviews [101]. The considered approaches are utilizing versatile opinions without providing further interpretation of them.

5.3 Summary of knowledge-based applications

A summary of the considered knowledge artifacts from the domain of quality evaluation is presented in Table 5.3. The table provides a classification of the artifacts in accordance to their application field and to the set of formulated questions.

Application	Task	State	View	Artifact	Knowledge
Technical fault detection and diagnosis	Monitoring Diagnosis	In use	Developer	Model, (prototype) KBS	Product, system behavior, symptoms, rules
Development process monitoring	Monitoring Diagnosis	Development	Developer	(Prototype) KBS	System behavior, normal patterns, rules
Product selection support	Assessment	Any	User	Prototype KBS	User needs, characteristics, rules or cases
User feedback analysis	Classification Assessment	In use	User	Prototype KBS	Characteristics, rules

Table 5.1: Summary of research on knowledge modelling for product evaluation

The following two conclusions can be derived from the conducted review:

- It can be observed that none of the considered artifacts corresponds to the defined goal of evaluating products being in operational use (column *State*) by combining both the developer and the user perspectives (column *View*).
- The closest area of research in terms of the task is FDD, as it aims at revealing quality issues of products. In contrast to such applications that focus on the technical performance of the products, with our research we aim at the user needs and consider both developer and user views for that.

With the next chapter we aim at closing the research gap and present a knowledge model for the target problem.

Chapter 6

Proposed knowledge model

Abstract. *In the following chapter we present the designed knowledge model for quality evaluation support. In order to construct the model we follow the framework described in chapter 4. First, we specify the tasks that the potential KBS should fulfil. Next, the knowledge model is defined based on these tasks and on the requirements derived from the domain exploration in chapters 2 and 3. Finally, the construction of an implementation-specific design model is outlined.*

6.1 Tasks definition

As defined in the introduction, the goal we follow is to support quality evaluation of products when they are in operational use. The evaluation should focus on the *actual user needs*, the output of the process should aid the developers in making their decisions on subsequent product improvement. This results in the following two tasks to be solved by the perspective KBS:

quality in use — To detect issues in *quality in use* of the products.

external quality — To describe the detected issues in terms of *external quality*.

The main principles are similar to the previously described principles from the FDD field: detecting some symptoms in the system operation and suggesting the root cause that they result from. Within our research the symptoms are some issues that can be detected when the products are being used. The fault is represented by the product non-conformity causing them. Once the symptoms are detected,

the potential KBS can create a hypothesis about this non-conformity in terms of external quality.
The tasks described above correspond to the monitoring and the diagnosis tasks defined in the CommonKADs design guidelines. According to the corresponding task templates, the KBS requires knowledge on the considered system and on its operation [34]. We identify and specify such knowledge in the following sections.

6.2 Knowledge model design

We start this section with identifying the relevant knowledge. Next, we specify the output that the potential KBS should deliver. Finally, we present the target knowledge model as a base for this KBS.

6.2.1 Knowledge identification

The knowledge about the considered system and its behavior can be acquired from various sources, for instance, from the domain experts, from documentation or by exploring the product. Since the monitoring task aims at the quality in use, we consider the outcome of the user-product interaction and focus on the following two required components:

interaction — We take the developer view on the interaction. The developer aims at providing a certain quality in use based on his understanding of the user needs. Consequently, he forms certain expectations regarding the product usage and can compare the actual usage with them. We define this as a *developer-perceived quality in use*.

user feedback — The user feedback is a subjective component expressing the user view of product quality, resulting from the quality in use. We therefore define it as a *user-perceived product quality*.

By defining the aforedescribed components we answer the second research question:

RQ2 Which *aspects of product usage* need to be considered within the evaluation?

The Figure 11 summarizes the views: the user compares the quality in use with his needs and provides a feedback on the product external quality. The developer indirectly evaluates the quality in use basing on his expectations.

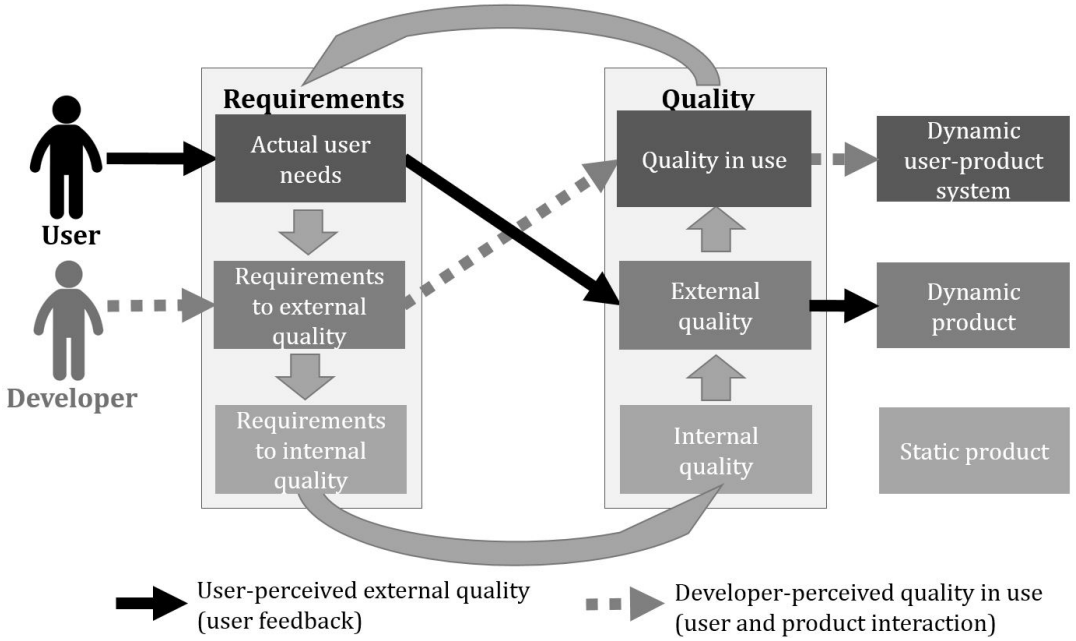

Figure 11: Target view of quality (modification of a figure from ISO/IEC 25010)

In the next section we formally specify the target output of the KBS and the knowledge that can be derived for that from the defined product usage aspects.

6.2.2 Knowledge specification

As the first step of knowledge specification we create a conceptual model demonstrating the key business-level objects and the main relations among them. The model can be found in Figure 12. The user-product interaction is represented by a sequence of actions. Each action involves a product function executed in some modality (REQ1) within some context (REQ3, REQ4).

Target output The *non-conformity* depicts the discrepancy between the product design and the user needs, which appears during the interaction. This difference results in the issues that are experienced by the users and can thus be

reflected in the user feedback and in the interaction itself. In accordance with the diagnosis task template, we define such knowledge pointing to the non-conformity as symptoms.

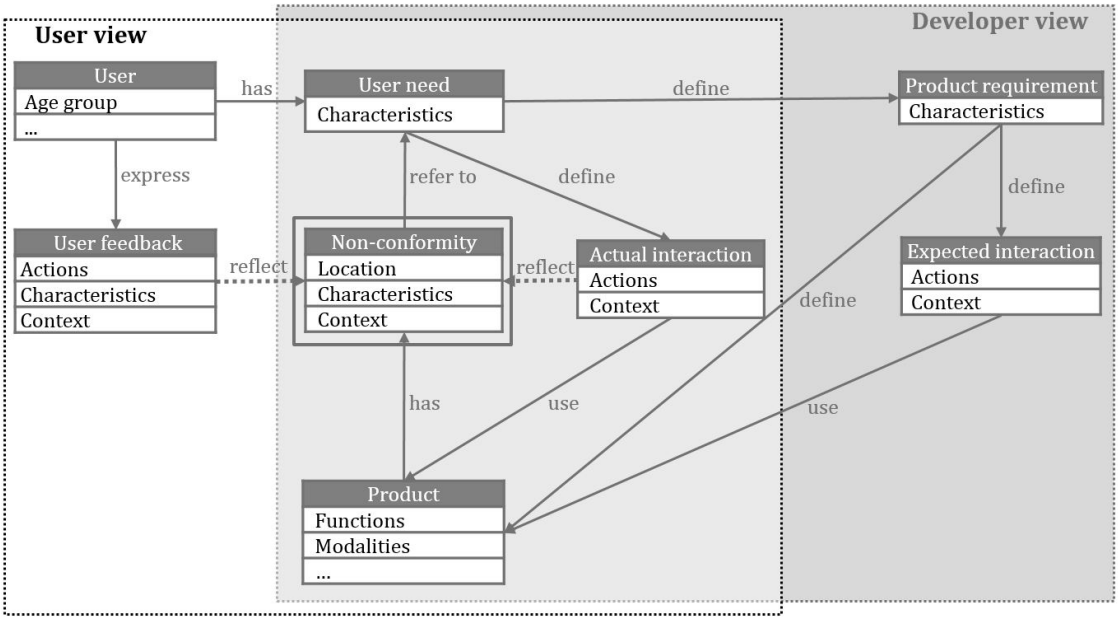

Figure 12: Conceptual model

The diagnosis task implies determining the fault type and as many other details as possible, such as the fault location or detection time. We thus define the non-conformity as a multidimensional concept described with the following knowledge:

characteristics — Product external quality characteristics causing the symptoms.

location — The location defining corresponding product part, for example, a particular function or modality.

context — The context in which the issues appear, related to the user, product or environment.

The defined knowledge comprehensively describes the non-conformity and facilitates its subsequent localization and elimination. By defining this concept we answer the first research question:

RQ1 How can *product non-conformities* to the *actual user* needs be defined?

In order to suggest the relevant product characteristics we adapt the ISO/IEC 25010 characteristics related to quality in use for the direct primary user [66]:

suitability Functional suitability defines how well product functions correspond to the actual user needs, i.e. cover all goals and provide correct outcome. We extend the characteristic to cover obsolete functions, i.e. functions that are provided, but not required by the users.

efficiency Performance efficiency defines product performance in relation to the required resources, e.g. to the product response time.

usability Usability defines how well the product can be used by the user to reach his goal with satisfaction. This includes ease of operation and control, protection from user error, product aesthetics, etc.

security Security defines the degree to which the product protects user data.

reliability Reliability defines how well the product performs the required functions under specified conditions. This includes product availability, fault tolerance and ability to recover from failures.

The characteristics above can be further decomposed into sub-characteristics to address the multi-level nature of quality (REQ7). In the next paragraph we present the knowledge model to enable KBS to suggest the above-defined non-conformities.

Designed knowledge model The proposed knowledge model consists of two main components: domain knowledge about the user-product system and the symptoms that can be fetched from the interaction and from the user feedback. A simplified model representation can be found in Figure 13.

User-product system. The knowledge about the system of the user, the product and the environment defines the location and the context of the issues. We introduce the following high-level taxonomy to represent the corresponding knowledge:

function Product functions organized hierarchically to reflect different granularity levels.

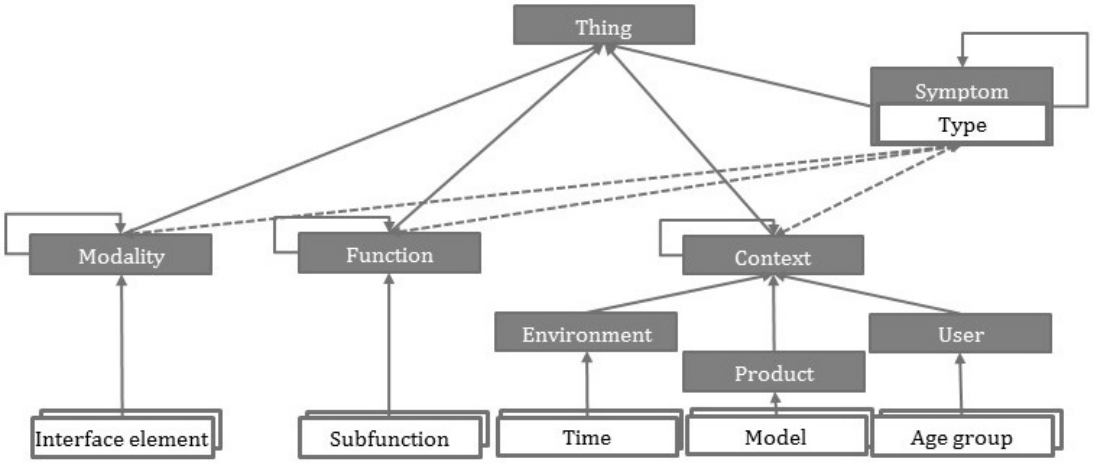

Figure 13: Proposed knowledge model (simplified)

modality — Product modalities presented hierarchically with particular interface elements at the lowest level.

product — Concepts defining other knowledge about the product, e.g. existing models or versions and other configurations.

user — Concepts capturing the user context: age group, experience with the same or similar products, etc.

environment — Concepts reflecting other contextual information, like geographic location or time.

The following properties are further defined over the described taxonomy:

function types — Properties differentiating between function types, like primary and secondary ones. The functions can be also assigned with the message they carry, e.g. notification on successful completion or on the occurrence of a technical problem.

related functions — Properties defining connections between related primary and secondary functions, such as settings. Functions serving the same goals, like sound adjustments, can be marked as equivalent or alternative.

function modality	Relations defining available modalities for the functions.
configurations	Relations further defining possible product configurations, for example, by assigning a list of integrated and optional functions or modalities.
time context	The time aspect for the concepts can be reflected via relations to the time context, e.g. to express the product age.
context limitations	Properties expressing existing environmental context limitations, like function unavailability for a specific location.

Symptoms. In the previous paragraph we presented a generic model of a user-product system. The symptoms we define in the following paragraph result from the defined quality viewpoints on this system. The goal is to model the knowledge that can point to the product non-conformity to the user needs.
The symptoms are represented with a corresponding concept designed to capture the knowledge from the user feedback and the user-product interaction. A symptom is defined through the following properties and relations to the previously defined entities at some granularity level:

characteristics	The product characteristics that the symptom is pointing to is defined with a corresponding symptom type. The relations between the type and the characteristics can be represented with causal knowledge.
location	Relations to the instances from function and/or modalities determine the location of the non-conformity. An example from the infotainment field is setting up a navigation goal using voice.
context	The context is defined with relations to the product, user and environment instances.

In order to fulfil the evaluation purposes the symptoms have to have a defined type and location expressed by least one relation to a function or modality. We now define possible symptom types for the defined input.
For the *user feedback symptoms* we consider the following two aspects:

ID	Feedback symptom type		Characteristics
	Sentiment	Content	
F1	No	No feedback	Suitability
F2	Negative	Functionality is missing, not found	Suitability
			Usability
			Reliability
F3		Product crashes, does not work properly	Efficiency
			Suitability
			Reliability
			Usability
F4		Product operation is slow	Efficiency
			Usability
F5		Product is not secure or confidential	Security
			Usability
F6		Function or modality is not needed	Suitability
			Usability
F7		Product is not visually attractive	Usability
F8	Negative/neutral	Product is inconvenient, unclear	Usability

Table 6.1: Symptoms from the user feedback

content The content provides the user direct or indirect interpretation of the product quality, expressed as a product characteristics deviating from his expectations in some context.

sentiment The sentiment defines user emotions. Positive experience of particular users in some context does not provide any value for the issues detection and diagnosis. We thus neglect positive feedback and focus on negative criticism or help requests from the user side.

The summary of the user feedback symptoms is presented in Table 6.1. For each type a differential of possible interpretations is provided, representing the causal diagnosis knowledge. This knowledge accounts for the feedback inaccuracy and defines external characteristics that could cause the user-experienced issue. For instance, a complaint about a missing function means that at least one of the user goals is not satisfied. The actual reason for that can be that the function is hard to find, is not recognized by the user or is not achievable due to technical problems

(F2). A confidentiality concern is a result of a user low trust in the product. The reason can reside in a lack of explanation provided to the user, rather than in insufficient security mechanisms (F5). An absence of any feedback indicates that the function is either not recognized as needed (F1) or that the user is satisfied with it.

For the definition of *user-product interaction symptoms* we formulate the developer expectations through the following principles:

goal achieved — The interaction with the product should end up with the user reaching his goal. The primary functions are therefore expected to be frequently used, other functions are to be used less often.

efficiency — The user goals have to be reached optimally, with minimal effort. This implies usage of the shortest paths, absence of cancelled steps, which can point to human errors, minimal time required for task completion, etc.

In Table 6.2 we define the interaction symptoms types and provide a list of external characteristics, which can cause them. For instance, a frequent pattern not ending up with completion of an assumed user goal (I3) can indicate that the user does not understand how to proceed, does not trust the product, follows another goal, as well as that the product acts too slowly or experiences technical problems. A complete absence of usage (I1), for example, can indicate that the user does not need the function or does not want to use it. Alternatively, he can be not aware of the function or the function can be not available due to technical problems.

The model provides further opportunities for the prioritization of possible non-conformities by combining the symptoms. A combination of a "no usage" (I1) and a "no feedback" (F1) symptoms could, for instance, assign a higher certainty to the functional suitability, implying that a function is not needed, rather than low usability, reliability or security.

In this section we have presented the designed knowledge model and answered the remaining research questions that are listed below:

RQ3 Which *relevant information* can be derived from analyzing product usage?

RQ4 How can *meaningful conclusions* be derived from the acquired information?

ID	Interaction symptom type	Characteristics
I1	No usage	Suitability
		Usability
		Security
		Reliability
I2	Rare usage	Suitability
		Usability
		Security
I3	Unsuccessful pattern (assumed user goal is not reached)	Usability
		Suitability
		Security
		Efficiency
		Reliability
I4	Non-optimal usage (user-product system acts inefficiently)	Usability
		Suitability
		Efficiency
		Reliability
I5	Unknown usage (the user goal is not clear to the developer)	Suitability
		Usability

Table 6.2: Symptoms from user-product interaction

6.3 Design model implementation

This chapter provides some details on developing the design model and defines a set of general guidelines for constructing a prototype or a full-scale KBS basing on the described knowledge model.

6.3.1 Selecting modeling formalism

We choose *ontologies* as a modeling formalism for the design model implementation due to the following reasons:

- The ontological models base on an open-world assumption and are well-suited for reasoning for new knowledge inference (REQ6).
- The ontologies provide effective mechanisms to maintain granularity (REQ2, REQ7) and other complex dependencies among different items (REQ4).

What is more, the models can be expanded with widely available reusable knowledge in an ontological form [85]. The context, for example, can be defined with existing time [102] or location schemata [103]. Some ontologies are designed specifically for particular domains, for example, for in-vehicle infotainment systems [104]. The causal knowledge on the relations between the symptoms and the characteristics that they point to can be represented in a form of rules.
In the next section we provide some guidelines on constructing and applying a KBS using the design model as a core component.

6.3.2 Applying the KBS

We provide a general minimal set of steps to be taken in order to utilize a KBS that can be built over the presented model. These steps are briefly described below:

1. Build a domain-specific knowledge base basing on the provided model.
2. Acquire symptoms from the interaction and from the user feedback. Save the symptoms in the knowledge base according to the model.
3. Check the knowledge base consistency and perform knowledge inference.
4. Evaluate the hypotheses received as a result of the inference.

The latter point is considered in more detail in the following section.

6.3.3 Evaluating the results

Reproducing the product non-conformities suggested by the KBS is challenging, because this can require the same context, e.g, the same users, as well as the same or similar environment. Other possible ways of evaluating the output include, but are not limited to, consulting domain experts or gathering an extra feedback from the users. As a result of the evaluation the suggested non-conformities can be divided into the following two categories:

false positives False positives are the non-conformities suggested by the KBS, but not confirmed by the evaluation. Such findings might serve as an

indicator that the knowledge system requires an improvement, for example, an update of the knowledge base.

true positives — True positives are the suggested non-conformities that were confirmed by the evaluation. Such findings can be used to plan the product improvement.

Detection of the existing non-conformities that were not suggested by the KBS can be performed by comparing the output to the results received by a human expert. In this chapter we have presented the knowledge model as a solution for the defined problem. While designing it, we have answered the research questions. Several guidelines were described to suggest an implementation of the design model and for constructing a KBS. In the next chapter we present a case study with an in-car infotainment system.

Chapter 7

Infotainment system case study

Abstract. *For the demonstration and evaluation purposes we design a prototype KBS. This chapter provides the implementation details and illustrates the application of the prototype on evaluation of an in-vehicle infotainment system. This is performed with the following three steps. First, we explain how the prototype is designed. Next, we define how the knowledge base for the infotainment system is created basing on the knowledge model and on the guidelines defined in the previous chapter. Finally, we present the prototype application by using available data on the system usage. The utilized data sources are social media and vehicles logfiles.*

7.1 Prototype implementation

This section presents implementation and application details of the KBS prototype. We provide some insights into the prototype development history, describe the resultant architecture and reveal the technologies we used.

7.1.1 History of prototype development

For the case study we select an in-vehicle infotainment system supported by one of the world largest automotive manufacturers and consequently widely used by customers across different countries. The complex product offers a broad variety of functions available in multiple modalities and utilizing various interface elements. Within the study we explore the system itself, as well as the potential data sources.

Along with official user manuals, we consider *logfiles* collected at the vehicle side. Such files represent recordings of user interaction with the infotainment system. The exploration also reveals existing software aimed at investigating the product usage based on the logfiles. The concept consists in calculating and graphically representing several statistical indicators, such as frequency of usage for different functions [105].
The tool is considered as a starting point for the prototype construction. The development is performed in several steps. We expand the list of the imported types of the logfile entries and implement a more extensive preprocessing to address several detected data quality problems. Wider capabilities for the exploratory analysis are provided by introducing new indicators, such as frequency of transitions between different functions or a number of used functions per trip [106]. As a source of user feedback we select *social media* messages and implement the required importing and preprocessing steps for the textual data.
In the next step, we integrate several operators to detect patterns from both logfiles and social media. The knowledge base component is created to store these patterns and the domain knowledge. The inference component is added to detect the symptoms among these patterns. In the following section we present the resultant technical architecture of the prototype.

7.1.2 Prototype architecture

The KBS prototype we design for the case study consists of the following four main components:

interface	The interface serves for providing the KBS with the domain knowledge and the data and receiving the inferred knowledge from it.
acquisition	The module created for acquiring the required knowledge from the provided input.
knowledge base	The module designed for storing the domain knowledge, the knowledge acquired from the data and the inferred knowledge.
inference	The module aimed at inferring previously unknown knowledge from the knowledge base.

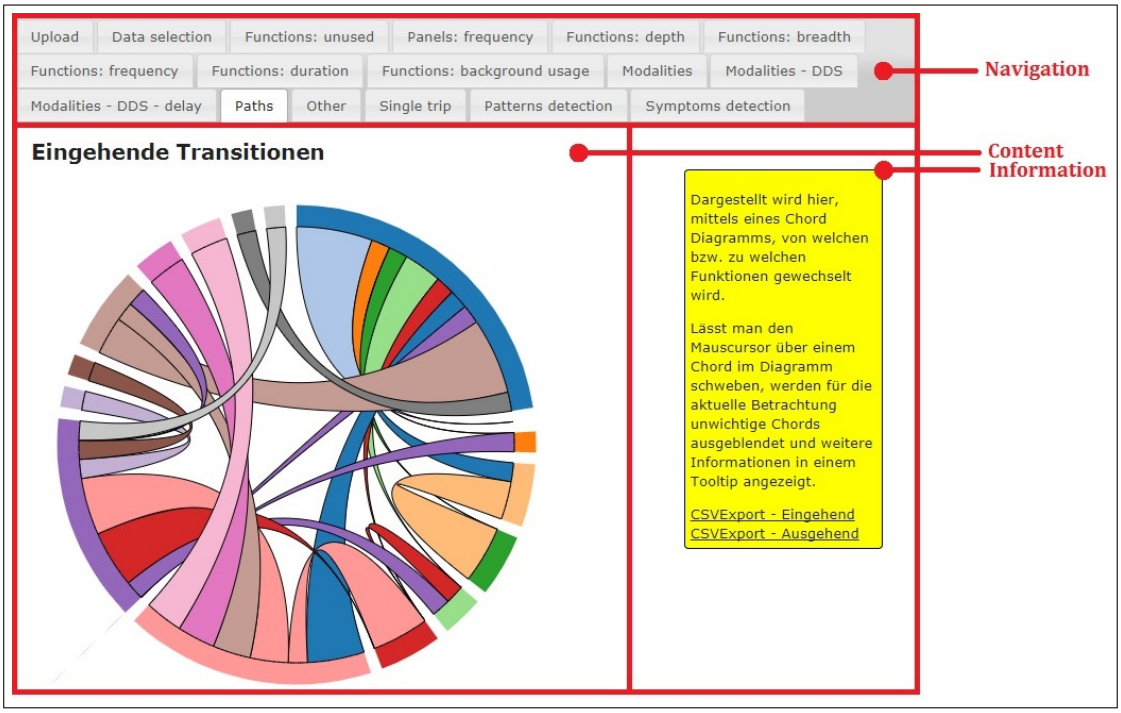

Figure 14: The overview of the propotype interface

In the following paragraphs we consider the aforementioned components in more detail to outline corresponding implementation aspects of them.

Interface. The prototype uses a web-based interface requiring a browser for the interaction. The used technologies include Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) for styling. The main components of the interface are schematically presented in Figure 14 and are listed below:

navigation — The navigation tabs serve for switching between the functional areas of the prototype, e.g. the area for providing the data input or for exploring statistical indications derived from it.

content — The area providing the main functional content and some interface elements for using the prototype.

information — The hints are delivering extra information to support the user in his interaction with the prototype.

Acquisition. The acquisition module utilizes different technologies to preprocess the input and to derive knowledge out of it. Below we outline the main subcomponents of the module together with the implementation details:

scripts JavaScript and AJAX are used for parsing and performing other preprocessing and exploratory steps over the input. For visually rendering the result the D3.js library[1] is used.

database Neo4j graph-based database[2] is selected for storing the parsed data. For the interaction with the database the Cypher query language is applied.

servlets Several Java-servlets are added for detecting patterns in the data. For some of them the RapidMiner[3] software development kit (SDK) is used.

container Apache Tomcat webserver[4] is selected as a servlet container.

The communication within the module bases on the REST API (Representational State Transfer Application programming interface). The exchange is performed in accordance with the JavaScript Object Notation (JSON). The module communicates with the knowledge base and the inference module using the OWL API[5].

Knowledge base. For implementing the knowledge model and preparing the knowledge base we use the Protégé tool[6] together with an SWRLTab Plugin[7]. The resultant ontology is expressed in OWL, the rules are created in SWRL.

Inference. The inference module utilizes the OWL2RL and Pellet [107] reasoners for deriving hidden knowledge from the knowledge base.

In the next section we describe the application logic of the prototype. The application on an actual in-car infotainment system is presented in subsection 7.3.

[1] Official web-site: d3js.org
[2] Official web-site: neo4j.com
[3] Data science platform. Official web-site: rapidminer.com
[4] Official web-site: tomcat.apache.org
[5] Official web-site: owlapi.sourceforge.net
[6] Ontology editor and framework. Official web-site: protege.stanford.edu
[7] Web-site: github.com/protegeproject/swrlapi/wiki

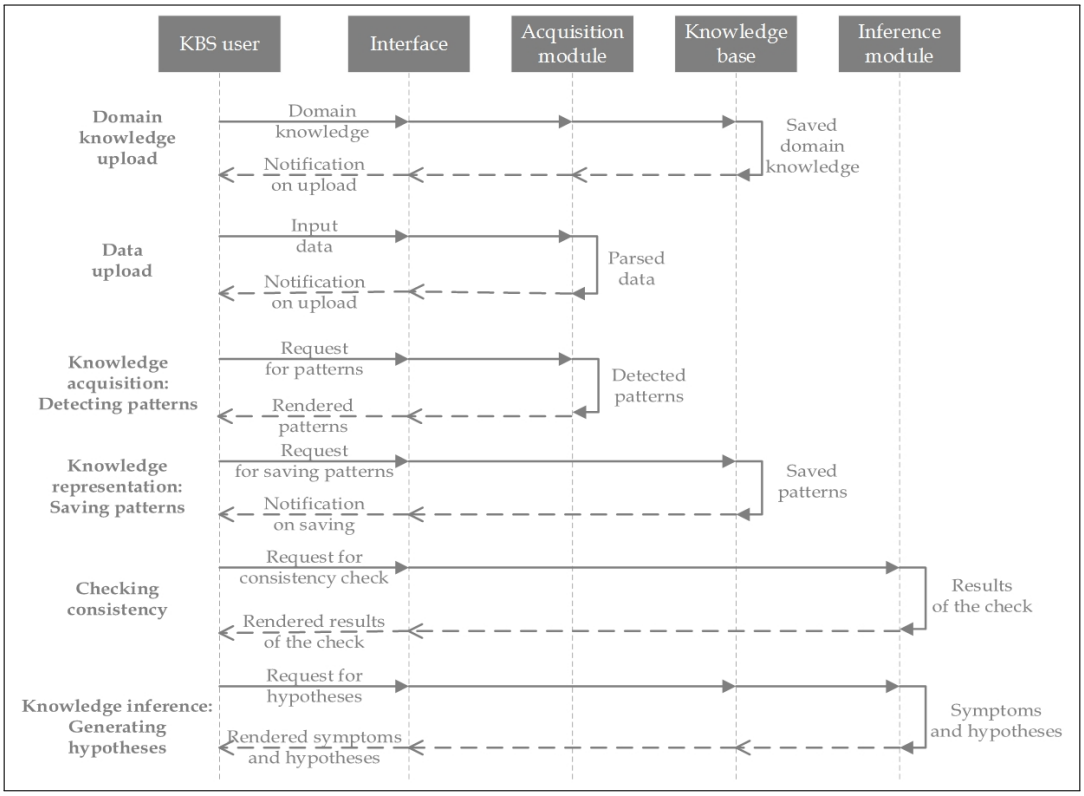

Figure 15: Sequence diagram for prototype application

7.2 Prototype application logic

The intended usage of the prototype is illustrated with a UML sequence diagram in Figure 15. The user and the aforedescribed components are depicted with vertical lines. The horizontal lines represent the communication among these actors. We show a combined representation of the main use cases. These are marked on the left side of the diagram and are further explained below. For the sake of simplicity we define only the normal flows representing the optimal sequences of steps. Each of the use cases is triggered by the user.

U1. Domain knowledge upload. As one of the first steps, the KBS has to receive the required domain knowledge. This is performed via the interface by the acquisition module and follows the same steps as the data upload (use case U2). The knowledge is saved in the database and the knowledge base.

U2. Data upload. The data on the user-product interaction and the user feedback are uploaded and preprocessed.

actors User, interface, acquisition module.

flow
1. The interface provides a dialogue to select locally stored data.
2. The user selects the data to upload and confirms the selection.
3. The acquisition module preprocesses the data and saves it in the database.
4. The interface notifies the user on successful saving of the data.

postcondition The parsed data is saved in the database.

The prototype supports various mechanisms to support exploration of the uploaded data, as an intermediate step. This includes visual representation of various statistical parameters, like the duration of usage for different product functions.

U3. Pattern detection. The process of knowledge acquisition from the provided data has a goal of detecting some patterns that can be potential symptoms.

actors User, interface, acquisition module.

precondition The parsed data is saved in the database.

flow
1. The user selects the data from the database and confirms the selection.
2. The user selects the pattern detection approach.
3. The acquisition module performs the pattern detection.
4. The interface returns the detected patterns to the user.

postcondition The detected patterns are presented to the user.

U4. Patterns saving. The knowledge acquired from the provided data is saved in the knowledge base.

actors User, interface, acquisition module, knowledge base.

precondition Some patterns are detected.

flow
1. The user selects the patterns to be saved in the knowledge base and confirms the selection.
2. The acquisition module represents the patterns in the knowledge base according to the knowledge model.
3. The interface confirms that the patterns are saved.

postcondition The patterns are saved in the knowledge base.

U5. Consistency check. Before starting the knowledge inference the user can check if the knowledge base is consistent.

actors User, interface, knowledge base, inference module.

flow
1. The inference component runs the consistency check.
2. The interface returns the results of the check to the user.

postcondition It is determined that the knowledge base is consistent.

U6. Symptom detection. Within the inference process some missing knowledge can be inferred, the stored patterns can be classified as symptoms and the resultant hypotheses should be returned to the user.

actors User, inference module, interface, knowledge base.

preconditions The knowledge base is consistent and includes the patterns.

flow
1. The inference module infers missing knowledge and launches reasoning to detect symptoms among the patterns.
2. The interface returns the results to the user.

postcondition The symptoms and the hypotheses are provided to the user.

Figure 16: A screenshot from the infotainment simulator

7.3 Infotainment system evaluation

In this section we illustrate the application of the described KBS prototype for the evaluation of an actual in-car infotainment system.

7.3.1 Knowledge base creation

Before applying the prototype the corresponding application-specific knowledge base has to be prepared. This section presents how the knowledge base for the actual in-vehicle infotainment system is built and implemented. We define the sources of knowledge used for that, explain how the domain knowledge from the sources is represented in the base and provide the rules designed for the case study.

Sources of domain knowledge. As previously mentioned, the selected infotainment system supports different modalities and interface elements. These are: voice operation, touchscreen, buttons next to the display, buttons on the steering wheel, etc. The list of provided functions includes, in particular, extensive smartphone integration, monitoring of the vehicle status, Wi-Fi support, as well as radio

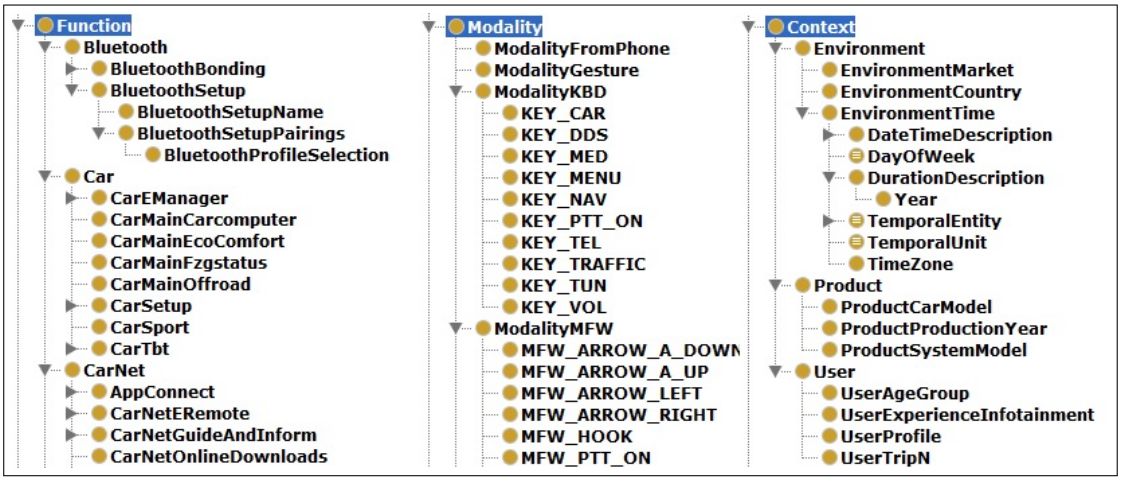

Figure 17: Building the knowledge base (defining taxonomy)

and audio playbacks and navigation. In order to acquire the knowledge on this complex system we use the following sources and methods:

actual system — Exploring different versions of the infotainment system in an operational environment.

simulator — Utilizing a tool simulating the behavior of the system. The simulator provides mockups of the system interface together with the corresponding internal names of the functions. An example of a mockup is provided in Figure 16.

documentation — Studying official manuals, as well as publicly available information provided by the developer on the Internet.

logfiles — Analysing the structure of the logfiles acquired by tracking the interaction with the system.

The required knowledge acquired from the aforementioned sources is described in the next subsection.

Domain knowledge engineering. The domain knowledge on the system functions, configurations, etc. is derived from the selected sources. The results are

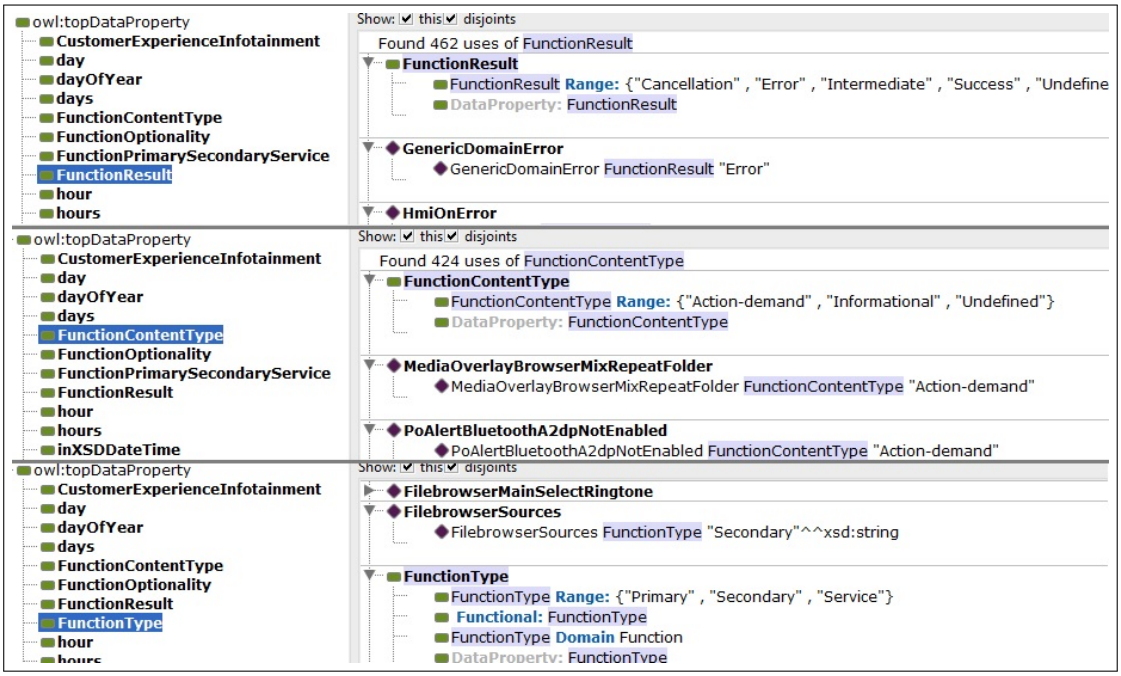

Figure 18: Building the knowledge base (defining properties)

represented in the ontology. The Figure 17 illustrates the first steps of the knowledge base generation, namely defining the taxonomy representing the knowledge on the user-product system.

We define the hierarchy of the functions and modalities. For example, a navigation function can be narrowed down to an address search or further down to a letter input. The hard key modality can be narrowed down to a group of buttons on the steering wheel or further down to a concrete button.

The Figure 18 illustrates some properties defined over the taxonomy, as the next step of the knowledge base generation. These properties are described below:

function types — The following data properties differentiating between the function types are assigned:

- *function type.* The property distinguishes primary and secondary functions, e.g. making calls and coupling a smartphone over Bluetooth. The service functions are not corresponding to the user goals.

- *result*. The functions are assigned with the message they carry, e.g. notification on successful coupling or on a technical problem occurrence. The range of the values include: Cancellation, Error, Intermediate and Success.

related functions — The properties defining related functions connect:

- *secondary and primary functions* related to each other, like the Bluetooth and the phone calls.
- *equivalent* or alternative functions serving the same goal, e.g. different ways of adjusting sounds settings.

function modality — The relations are created to define available modalities for the functions: the buttons located on the steering wheel are linked to making calls, but are unavailable for Internet browsing.

configurations — Other relations further describing possible product configurations are defined. The vehicle models are mapped to the supported infotainment system versions, while the system versions are assigned to a list of integrated and optional functions and modalities. For example, the FM-radio is present in all versions, while the voice operation is optional.

time context — The time aspect is reflected via relations to the time context, e.g. to define the infotainment system age.

context limitations — The properties expressing existing environmental context limitations are created, like function unavailability for a specific location.

We create an ontology class for the symptoms and define properties relating it to other concepts, e.g. to specify which infotainment system model they refer to. The Figure 19 illustrates a fragment of the resultant ontological knowledge base containing the relevant knowledge on the infotainment system acquired from the defined sources.

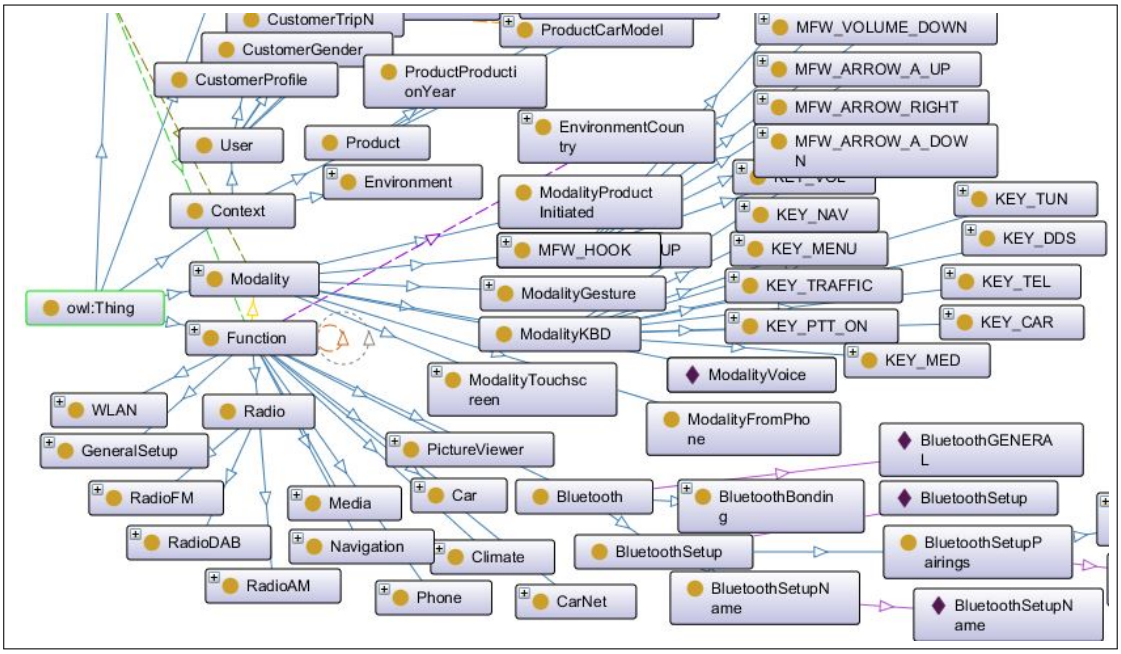

Figure 19: Fragment of the infotainment domain ontology

Generating inference rules. As the final step of the knowledge base construction, the expectations regarding the user-product interaction are formalized based on the defined principles. A set of rules for symptoms detection is created. Below we provide some examples of such rules:

No usage — The rules for the *no usage* (I1) symptom are application-independent. For the functions they require the following two conditions to be true: the usage frequency has a value of 0 for some users and the function should primary. For the modalities usage only the first condition has to be satisfied.

Rare usage — For the *rare usage* (I2) we define a rule detecting functions and modalities that have a usage frequency greater than 0, but less than expected. The expectations vary for the primary and the secondary functions.

Unsuccessful — The *unsuccessful* symptoms (I3) include the following patterns: the patterns containing a function with the *result* property equal to *error*

and the patterns showing a usage frequency greater than expected for a secondary function.

Non-optimal The *non-optimal* symptoms (I4) cover the patters containing a function with the *result* property equal to *cancellation* and the patterns including "backward" steps, i.e. some actions cancelling the previous steps.

7.3.2 Knowledge acquisition from the sources

This section describes the available sources on the infotainment system usage, the steps that were performed for their analysis, as well as the resultant output.

Logfiles for interaction analysis

The acquisition of knowledge on the user-product interaction is performed with the following steps: data understanding, preprocessing, modeling, interpretation and evaluation of the results. In the following paragraphs we describe the used data source and define the steps in detail.

Logfiles as a source of knowledge. In the logfiles the actions taken by the two parties are recorded. Such files are generated at the vehicle side, the data is collected on an external storage. The raw data is converted to the CSV (Comma-separated values) format and can be then analysed for different purposes. The procedures of data generation, acquisition and converting are beyond the scope. An example of the logfile can be found in Figure 20. Each of the entries includes the following fields:

type The type of the entry shows the category of the occured event: a radio station change, volume reduction or displaying a panel on the screen.

date The day and time of the event occurrence are tracked.

timerId Id is received by a set of logfile entries.

ticks Number of milliseconds passed from assigning a new timerId.

type	date	timerId	ticks	data
timer	14.02.2014 20:31:52	002	34537	<timer> DateStamp: 1392409947000 swRelease: 0137 Partnumber: 3Q0035864 SerNumber: VWZAZ2R8539540

```
usb,14.02.2014 20:32:25,002,33453, <usb> State: Inserted Slot: 1 VendorId: 1060 ProductId: 16448 Class: 8 SubClass: 6 Protocol: 80 Speed
misc,14.02.2014 20:32:25,002,33470, <misc> AppInfo: tsd.mibstd2.system.itr.main MsgType: Info Msg: Diag: Itr enabled.
env,14.02.2014 20:32:27,002,34559, <env> Voltage: 122 Temp411Int: 16 Temp411IntMin: 2 Temp411IntMax: 52 Temp411Ext: 14 Temp411ExtMin: 1
display,14.02.2014 20:32:27,002,34566, <display> Voltage: 123 Temp: 75 TempMax: 84
kbd,14.02.2014 20:32:27,002,34575, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_PRESSED
kbd,14.02.2014 20:32:27,002,34727, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_RELEASED
tuner,14.02.2014 20:32:27,002,34767, <tuner> IRQ: SetStation- 89000 PI:d0db
kbd,14.02.2014 20:32:53,002,60945, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_PRESSED
kbd,14.02.2014 20:32:53,002,61099, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_RELEASED
tuner,14.02.2014 20:32:53,002,61144, <tuner> IRQ: SetStation-100300 PI:d385
kbd,14.02.2014 20:33:08,002,75992, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_PRESSED
kbd,14.02.2014 20:33:08,002,76162, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_RELEASED
tuner,14.02.2014 20:33:08,002,76201, <tuner> IRQ: SetStation- 92100 PI:d382
kbd,14.02.2014 20:33:20,002,88359, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_PRESSED
kbd,14.02.2014 20:33:20,002,88499, <kbd> keyboardID: KBD_TOUCHSCREEN_FRONT keyID: KEY_DDS keyState: KST_RELEASED
tuner,14.02.2014 20:33:20,002,88531, <tuner> IRQ: SetStation- 89900 PI:d383
```

Figure 20: Example of the logfile in CSV format

data — The main message carried by the logfile entry. The field represents the event that occurred, e.g. the name of the panel that was shown.

For the case study the data on the interaction with the infotainment system was voluntary provided by 10 proband participants. The participants belonged to different age groups and had different backgrounds. In particular, the experience with infotainment systems spanned from "No" to "Expert" [108].
All infotainment systems had default factory settings at the beginning of the survey. The interaction tracking was started shortly after the delivery of the vehicles. The overall period of recording was two weeks for each participant. During this period the logfiles readout to an external storage was performed three times [108]. The survey participants had no pre-defined tasks. The vehicles and the infotainment systems were used in an operational environment. The corresponding car and infotainment system model, as well as the time and the geographic region of the case study are known.

Logfile preprocessing. The preprocessing of the logfiles is required due to the following reasons:

purpose — As the logfiles were originally designed for tracking such technical problems like crashes and others, the data structure needed an extensive preprocessing to make it suitable for the purpose of the analysis.

buffer — The tracking of the interaction on the vehicle side utilizes a ring buffer with a limited maximum number of entries for different logfile types.

Once the upper limit is reached, the buffer gets an overflow and the logfiles start being overwritten by new entries.

delay — The recording after the car ignition is starting with a delay.

timestamp — The timestamp recorded in the logfiles corresponds to the time settings of the infotainment system. This leads to incorrect data when the settings are not properly adjusted and to a confusion when the user changes the time settings during the trip.

The upload of the logfiles follows the use case *U2*. The user selects the files in the CSV format via the KBS interface. The system performs preprocessing steps aimed at preparing the data for pattern detection. These steps are described below:

1. *Filter* out irrelevant log entries. Since the logfiles were designed for catching technical problems, there is a number of logtypes not relevant for the analysis.
2. *Separate by trips.* The events are grouped by the trips taken by the driver.
3. *Remove* overlapping instances. The corrupted trips where the first entries are overwritten due to the buffer overflow are filtered out.
4. *Add artificial logfiles* for the cases when the start of the trip is lost. For example, when the users turn the car off, but leave the radio on, it can be deducted that next time the car is on the radio turns on.

The files are mapped to the domain knowledge and saved in the database. The list of the uploaded data files is presented to the user.

Detecting symptoms from logfiles. The process of detecting symptoms from the lofgiles follows the use cases *U3* - *U6*. The aim is to detect as many symptoms as possible. As the first step we perform pattern detection from the data. The first algorithm we apply is FP-growth [109] to mine frequent combinations of events during one trip. The alternative approach uses GSP-algorithm [110] for the sequential analysis preserving the order of these events. The approaches result in around 500 patterns. These are saved in the knowledge base, corresponding relations to the utilized functions, modalities, vehicle and infotainment system models,

as well as to the country of the experiment are established.
As the second step we apply the rules from the knowledge base to detect the symptoms among the patterns. The findings mainly refer to no or rare usage of functions and to unsuccessful patterns ending up with technical exceptions. Some of the examples are provided below:

Mirrorlink — Rare or no usage. The function was launched 5 times by 3 users, the average duration of usage was 70sec.

Picture Viewer — Rare or no usage. The function was launched 4 times by 3 users, in 3 times the duration of usage did not exceed 10 sec.

Bluethooth — Unsuccessful pattern. The Bluetooth coupling of a smartphone with the system failed.

Activation — Unsuccessful pattern. A completion of an online-activation for one of the system functions failed.

Social media for feedback analysis

The analysis of the user feedback basing on the social media messages is as well performed in the following steps: data understanding, preparation, modeling, interpretation and evaluation of the results. We describe each of the steps below.

Social media as a source of knowledge. For the case study we use a set of English and German textual messages from social media referring to the infotainment system. The crawling and initial preprocessing of the relevant data was performed by a third party. The corresponding procedures are beyond the scope. The data was collected for over two-year period and comprised around 1800 messages from various resources, mainly from automotive forums. When possible, the messages were assigned with the discussed product characteristics, with the system and the vehicle model, as well as the corresponding country. Sentiment analysis was performed to select negative and neutral messages.
An example of the social media messages can be found in Figure 21. Every entry includes the following fields:

ID	Date	Text	Title	Url	Germany	Norway	...	Navigation, negative	Navigation, p	...	
1440015	31.03.2015 22:28	Same for me,	Crackling sc	http://elbilfor	0	0	0	0	0	0	
1439974	31.03.2015 21:47	I experienced	Error in nav	http://elbilfor	0	0	0	1	0	0	
1440004	31.03.2015 14:06	A strange thin	Crackling sc	http://elbilfor	0	0	0	0	0	0	
1439928	31.03.2015 10:26	No problem t	Pre-heating	http://elbilfor	0	0	0	0	0	0	
1439395	31.03.2015 08:24	found this	Pre-heating	http://elbilfor	0	0	0	1	0	0	
1439408	31.03.2015 08:16	Hi! In our con	Error in nav	http://elbilfor	0	0	0	1	0	0	

Figure 21: Example of the data from social media

id Unique ID for each message collected from the Internet.

date Day and time according the information in the Internet.

text The main user message in the textual form.

title The title of the message.

url The URL-address of the message in the Internet.

tags A set of flags manually assigned by the third party.

Social media messages preprocessing. The upload and the preprocessing of the data from social media follows the use case *U2*. The preparation includes the following steps:

1. *Case folding* reducing all messages to a lowercase.
2. *Common misspellings correction* merging different forms of the same word using language-specific dictionaries.
3. *Shrinking acronyms* replacing full forms of the specific infotainment-related words to the corresponding acronyms. This is performed by creating language-specific dictionaries.
4. *Removal of useless specific knowledge structure*, like hashtags, URLs, etc.
5. *Tokenization* breaking messages into tokens using whitespaces and punctuation as delimiters.
6. *Stemming* conflating inflected forms of one word by reducing them to a common form.

7. *Removal of meaningless tokens* is performed in two steps: dropping general stopwords using a dictionary common for all domains (e.g. words *the*, *is*, *at*) and removing domain-specific stopwords.

8. *Vector representation.* Finally, the messages are represented in a vector-based binary token-occurrence form: every token from the data set receives a 1 value for every message in which it is present and 0 for all other messages.

The preprocessed data is used for further analysis.

Detecting symptoms from social media messages. The symptoms detection from social media is as well performed using two different approaches. With the FP-Growth algorithm we aim at detecting common combinations of terms utilized within the data set. Such combinations have proven to be meaningful indicators of common topics discusses in the data sets related to a limited technical field [111]. Additionally we utilize the *Lift* measure:

$$Lift(X \rightarrow Y) = \frac{Sup(X,Y)}{Sup(X)Sup(Y)}$$

where X and Y are two terms and *Sup* is the absolute number or ratio of messages containing the combination of these terms to the whole number of messages. We select the maximal combinations and filter out combinations with a low *Sup* and a *Lift* less or equal to 1.
With the clustering approach we aim at grouping semantically-related messages. For the generation of semantic clusters the K-means++, K-medoids, DBSCAN and Hierarchical Clustering algorithms are compared. The quality of the results is evaluated by the Silhouette Coefficient. In case of K-medoids the messages can be clearly assigned to suitable clusters [112]. The topics of the single groups are extracted, characterised by metadata and visualised with wordclouds. The results are manually interpreted, the flags assigned to the original messages are utilized to map them to the domain knowledge.
With the analysis we detect around 30 patterns that are saved in the knowledge base as symptoms of corresponding types. Some examples are presented below:

MirrorLink The number of supported applications is too limited.

MirrorLink	The applications stop working when the vehicle is moving.
MirrorLink	Some smartphones are not compatible with the system.
MirrorLink	The activation procedure for the function is not clear.
Picture viewer	There are no mentions of the functions.
Bluethooth	Frequent acoustic problems are experienced during telephone calls.
Navigation	The radio sound is not reduced during the navigation announcements.

7.3.3 Knowledge inference and evaluation

Each of the detected symptoms points to a set of characteristics that could have caused it. By combining them we can prioritize the hypotheses. For instance, the logfile analysis shows a no or rare usage of picture viewing (I1 and I2). The social media analysis reveals no common mentions of this function (F1). A combination of the symptoms can further give a priority to suitability as possible non-complying characteristic, meaning that the function is obsolete and is not recognized as needed by the users. The improvement steps would be to make the function more obvious and attractive or eliminate it from the further versions.
The second example refers to sound settings of a navigation function. The feedback analysis reveals a frequent user concern regarding the navigation announcements not lowering the radio volume. According to the user view, the volume reduction function is missing (F2). The interaction analysis shows that this function has never been used (I1). One of the prioritized hypothesis would be low usability meaning that the users are not aware of the volume settings, do not understand how to use or where to find them. The improvement could make the announcement adjustments easily accessible and understandable. Alternatively, the suitability could be improved by changing the default volume settings. The KBS therefore can encounter different viewpoints of the product and help to mutually strengthen the findings from the sources.
The participant of the survey were asked questions after the case study in order to evaluate the hypotheses. Some of the results are presented below:

Picture viewer	Participants confirmed no need for the function for pictures viewing on the infotainment screen.
Bluetooth	Some participants confirmed acoustic problems during the telephone calls conducted with the help of the system.
Bluetooth	Some participants confirmed problems with Bluetooth coupling of smartphones.
Activation	Several participants confirmed difficulties with activating one of the system functions online.

Therefore, multiple conclusions suggested by the KBS prototype were confirmed during the case study. The major findings were used to derive suggestions for further improvement of the infotainment system.

Chapter 8

Conclusion and future work

In this thesis we have introduced a domain knowledge model to support the evaluation of computer-based products in use with regards to the user needs. The relevance of the research has been illustrated by a literature review that revealed a corresponding gap in knowledge modeling. We designed the model by analysing the main characteristics of user-product system, considering it from the quality perspective and defining the target evaluation goal.
The resultant model incorporates both the developer and the user views on the product. We have defined the user-product model, as well as the knowledge that can be acquired from the user feedback and the user-product interaction. A perspective application of the model for a KBS design was outlined.
While constructing the model, we have answered the four research questions. Below we provide a summary of the derived answers:

RQ1 How can the *product non-conformities* to the *actual user* needs be defined?
The non-confomity is defined as a multidimensional concept, represented by a combination of product characteristics, location and context.

RQ2 Which aspects of the product usage need to be considered within the evaluation?
We define user feedback and user-product interaction as two crucial components to be considered during the evaluation process.

RQ3 Which relevant information can be derived from the defined input?

The relevant information is defined as a set of symptoms that can be extracted from the data on the user-product interaction and the user feedback.

RQ4 How can meaningful conclusions be derived from the acquired information? The conclusion can be represented as an interpretation of the symptoms in terms of the external quality.

The usage of the presented model allows to represent the disparate data from different sources in a form more suitable for further processing, deriving valuable insights and supporting decisions on product improvement. The model can serve as a major step for building a knowledge base and implementing inference procedures for semi-automated detection and diagnosis of quality issues. By basing on the common quality standards and on a generic user-product model we ensure the applicability to a wide range of computer-based products.

Since the model represents a succinct schema for quality evaluation support, a number of steps can be taken in order to extend it for every specific use case. The decision on the required adjustments is to be made basing on the requirements to the evaluation and on the data availability.

For the future research we suggest deriving specific guidelines on the model utilization within different domains. Moreover, more case studies can be conducted. For instance, the model application can be demonstrated and evaluated on other products, like smartphones or tablets. Alternative data sources can be involved, like customer surveys or observational data on user behavior. A similarity-based matching of the symptoms detected from the sources is to be integrated. Finally, a fully-implemented KBS is to be built and tested in an operational environment.

Appendix A

Ontologies components and design principles

In the following we consider in detail the structure of ontologies and their main design principles. The central components of ontologies include classes, instances, as well as data and object properties. These components are defined as follows:

classes The classes describe the domain concepts. Each class can be broken down to subclasses to represent the concept more specifically.

instances The individuals represent the instances of the defined classes. An ontology with a set of defined instances constitutes a knowledge base.

properties The properties defined over the entities describe their features. The object properties determine relations to other entities, the datatype properties are assigned with some data value.

Noy and McGuinness suggested the following seven common steps for the ontology design [85]:

1. Determine the domain that the ontology should cover, its goal and the potential users.
2. Consider reusing already existing ontological models.
3. Enumerate the important terms for the ontology.

4. Define the ontology classes and their hierarchy.
5. Define the properties of these classes to describe their internal structure.
6. Define the domain and the range of the properties.
7. Create individual instances for the classes.

The following three fundamental principles for the ontology design were defined:

- There is no single correct way to model a domain.
- The design process should be iterative.
- The ontology should be close to the physical or logical concepts.

Gruber defined the following criteria for the ontology design [113]:

Clarity	The definitions provided by the ontology should be objective and, when possible, complete.
Coherence	The inferences allowed by a ontology should comply with the definitions.
Extendibility	It should be possible to define new terms basing on an existing ontology vocabulary without revising it.
Bias	The ontology should be specified at the knowledge level with no dependency on the encoding at the symbol level.
Minimality	The ontology should make as few claims as possible by defining only the essential terms.

List of Tables

List of Figures

List of acronyms

API Application programming interface

CI Continuous (continual) improvement strategy

CSV Comma-separated values

FDD Technical fault detection and diagnosis

HMI Human-machine interaction

IS Information system(s)

KBS Knowledge-based (knowledge) system(s)

OWL Web Ontology Language

PDCA Plan-Do-Check-Act

PDSA Plan-Do-Study-Act

SWRL Semantic Web Rule Language

JSON JavaScript Object Notation

Bibliography

[1] "Number of smartphone users worldwide from 2014 to 2020 (in billions)," study report, eMarketer, 2016.

[2] "E-book sales as a percentage of total book sales worldwide in 2013 and 2018," study report, PricewaterhouseCoopers (PwC), 2014.

[3] "Connected car forecast: Global connected car market to grow threefold within five years," study report, GSMA, London, UK, 2013.

[4] G. Meixner, C. Häcker, B. Decker, S. Gerlach, A. Hess, K. Holl, A. Klaus, D. Lüddecke, D. Mauser, M. Orfgen, M. Poguntke, N. Walter, and R. Zhang, "Retrospective and Future Automotive Infotainment Systems—100 Years of User Interface Evolution," in *Automotive User Interfaces* (G. Meixner and C. Müller, eds.), pp. 3–53, Cham: Springer International Publishing, 2017.

[5] "U.S. Vehicle dependability study (VDS)," study report, J.D. Power, 2016.

[6] D. L. Strayer, J. M. Cooper, J. Turrill, J. R. Coleman, and R. J. Hopman, "Measuring Cognitive Distraction in the Automobile III: A Comparison of Ten 2015 In-Vehicle Information Systems," study report, AAA Foundation for traffic safety, Washington, DC, USA, 2015.

[7] M. Christel and K. C. Kang, "Issues In Requirement Elicitation," Technical Report CMU/SEI-92-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1992.

[8] "ISO/IEC FDIS 9126-1:2000. Information technology — Software product quality," final draft of standard, International Organization for Standardization (ISO), Geneva, Switzerland, 2000.

[9] N. Bevan, "Using the common industry format to document the context of use," in *International Conference on Human-Computer Interaction*, pp. 281–289, Springer, 2013.

[10] N. Bevan, J. Carter, and S. Harker, "ISO 9241-11 Revised: What Have We Learnt About Usability Since 1998?," in *Human-Computer Interaction: Design and Evaluation*, vol. 9169 of *Lecture Notes in Computer Science*, (Los Angeles, USA), pp. 143–151, Springer, 2015.

[11] S. Wagner, "Quality models," in *Software Product Quality Control*, pp. 29–89, Berlin: Springer, 2013.

[12] K. Ishikawa, *What Is Total Quality Control? The Japanese Way*. Englewood Cliffs, USA: NJ: Prentice-Hall, Inc, 1985.

[13] C. Kaiser, S. Alexander, and M. Fellmann, "Stocker A, Kaiser C, Fellmann M (2017) Quantified Vehicles. Novel Services for Vehicle Lifecycle Data," *Business and Information Systems Engineering*, pp. 125–130, 2017.

[14] K. Schneider, *Experience and Knowledge Management in Software Engineering*. Springer-Verlag Berlin Heidelberg, 2009.

[15] R. G. Dromey, "A model for software product quality," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995.

[16] J. Nanda, T. W. Simpson, S. R. T. Kumara, and S. B. Shooter, "A Methodology for Product Family Ontology Development Using Formal Concept Analysis and Web Ontology Language," *Journal of Computing and Information Science in Engineering*, vol. 6, no. 2, pp. 103–113, 2006.

[17] S. Rachuri, Y.-H. Han, S. Foufou, S. C. Feng, U. Roy, F. Wang, R. D. Sriram, and K. W. Lyons, "A Model for Capturing Product Assembly Information," *Journal of Computing and Information Science in Engineering*, vol. 6, no. 1, pp. 11–21, 2006.

[18] P. Witherell, S. Krishnamurty, and I. R. Grosse, "Ontologies for Supporting Engineering Design Optimization," *Journal of Computing and Information Science in Engineering*, vol. 7, no. 2, pp. 141–150, 2007.

[19] J. Rockwell, I. R. Grosse, S. Krishnamurty, and J. C. Wileden, "A Decision Support Ontology for collaborative decision making in engineering design," in *2009 International Symposium on Collaborative Technologies and Systems*, (Baltimore, MD, USA), pp. 1–9, IEEE, 2009.

[20] I. Jureta, J. Mylopoulos, and S. Faulkner, "Revisiting the Core Ontology and Problem in Requirements Engineering," in *16th IEEE International Requirements Engineering Conference*, (Barcelona, Spain), pp. 71–80, IEEE, 2008.

[21] Y. Elyusufi, H. Seghiouer, and M. A. Alimam, "Building profiles based on ontology for recommendation custom interfaces," in *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS)*, (Marrakech, Morocco), pp. 558–562, IEEE, 2014.

[22] T. J. Hagedorn, S. Krishnamurty, and I. R. Grosse, "An information model to support user-centered design of medical devices," *Journal of Biomedical Informatics*, vol. 62, pp. 181–194, 2016.

[23] B. Boehm, H. Kaspar, J. Brown, and Lipow, Myron, *Characteristics of Software Quality*. TRW, Amsterdam, The Netherlands: North Holland Publishing, 1978.

[24] S. Wagner, K. Lochmann, L. Heinemann, M. Klas, A. Trendowicz, R. Plosch, A. Seidi, A. Goeb, and J. Streit, "The Quamoco product quality modelling and assessment approach," in *Proceedings of the 34th International Conference on Software Engineering*, (Zurich, Switzeland), pp. 1133–1142, IEEE, 2012.

[25] B. Magoutas, C. Halaris, and G. Mentzas, "An Ontology for the Multi-perspective Evaluation of Quality in E-Government Services," in *Electronic Government* (M. A. Wimmer, J. Scholl, and A. Grönlund, eds.), vol. 4656, pp. 318–329, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[26] F. Corradini, F. de Angelis, A. Polzonetti, B. Re, and E. Brugnoni, "E-GOVQOS: An ontology for quality of e-government services," in *Proceedings*

of the 5th International Conference on Electronic Government, vol. 4084 of *Lecture Notes in Computer Science*, (Krakow, Poland), pp. 171–178, Springer, Heidelberg, 2006.

[27] G. Fischer, "User modeling in human–computer interaction," *User modeling and user-adapted interaction*, vol. 11, no. 1-2, pp. 65–86, 2001.

[28] Z. Obrenovic and D. Starcevic, "Modeling multimodal human-computer interaction," *Computer*, vol. 37, no. 9, pp. 65–72, 2004.

[29] B. Kitchenham, "Procedures for performing systematic reviews," Joint Technical Report 0400011T.1, Keele University and National ICT Australia Ltd., Keele, UK, Eversleigh, Australia, 2004.

[30] M. R. H. Iman and A. Rasoolzadegan, "Quantitative evaluation of software usability with a fuzzy expert system," in *Proceedings of the 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, (Mashhad, Iran), pp. 325–330, IEEE, 2015.

[31] B. Akinnuwesi, F.-M. E. Uzoka, and A. O. Osamiluyi, "Neuro-Fuzzy Expert System for evaluating the performance of Distributed Software System Architecture," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3313–3327, 2013.

[32] R. Isermann, "Model-based fault-detection and diagnosis – status and applications," *Annual Reviews in Control*, vol. 29, no. 1, pp. 71–85, 2005.

[33] A. Dix, F. Janet, G. Abowd, and R. Beale, *Human-Computer Interaction*. New York, USA: Prentice Hall, 2003.

[34] G. Schreiber, ed., *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, Mass: MIT Press, 2000.

[35] K. Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2008.

[36] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. Hillsdale, USA: L. Erlbaum Associates Inc, 1983.

[37] H. Bunt, "Issues in multimodal human-computer communication," in *International Conference on Cooperative Multimodal Communication*, pp. 1–12, Springer, 1995.

[38] J. Rasmussen, "Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and other Distinctions in Human Performance Models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 3, pp. 257–266, 1983.

[39] A. N. Leontiev, "Activity and consciousness," *Voprosy filosofii*, vol. 12, pp. 129–140, 1972.

[40] S. K. Card, T. P. Moran, and A. Newell, "The keystroke-level model for user performance time with interactive systems," *Communications of the ACM*, vol. 23, no. 7, pp. 396–410, 1980.

[41] A. Meroth and B. Tolg, eds., *Infotainmentsysteme im Kraftfahrzeug: Grundlagen, Komponenten, Systeme und Anwendungen*. Vieweg Praxiswissen, Wiesbaden, Germany: Vieweg, 1 ed., 2008.

[42] J. Rasmussen and K. Vicente, "Coping with human error," *International Journal of Man-Machine Studies*, vol. 31, no. 5, pp. 517–534, 1989.

[43] J. Coleman, *The Foundations of Social Action*. Cambridge, UK: The Belknap of Harvard University Press, 1990.

[44] H. A. Simon, *The Sciences of the Artificial*. Cambridge, UK: MIT Press, 3 ed., 1996.

[45] W. T. Singleton, *The Mind at Work. Psychological Ergonomics*. Cambridge University Press, 1989.

[46] J. Senders and N. Moray, *Human Error: Cause, Prediction and Reduction*. NJ, USA: Lawrence Erlbaum Associates Inc, 2008.

[47] N. Bevan, "Usability is Quality of Use," in *Advances in Human Factors/Ergonomics*, vol. 20, pp. 349–354, Elsevier, 1995.

[48] K. Krippendorff, *The Semantic Turn: A New Foundation for Design*. Boca Raton, USA: Taylor & Francis Ltd, 2005.

[49] L. von Bertalanffy, *General System Theory. Foundations. Development, Applications*. George Braziller Inc, 1968.

[50] "ISO 9241-210:2010. Ergonomics of human-system interaction Part 210: Human-centred design for interactive systems," standard, International Organization for Standardization (ISO), Geneva, Switzerland, 2010.

[51] B. Shneiderman and C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Boston, USA: Pearson/Addison Wesley, 4 ed., 2004.

[52] S. J. Fenves, S. Foufou, C. Bock, and R. D. Sriram, "CPM2: A Core Model for Product Data," *Journal of Computing and Information Science in Engineering*, vol. 8, no. 1, 2008.

[53] F. Quek, D. McNeill, R. Bryll, S. Duncan, X.-F. Ma, C. Kirbas, K. E. McCullough, and R. Ansari, "Multimodal human discourse: Gesture and speech," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 9, no. 3, pp. 171–193, 2002.

[54] A. Jaimes and N. Sebe, "Multimodal Human Computer Interaction: A Survey," in *Computer Vision in Human-Computer Interaction. Proceedings of Workshop on Human Computer Interaction in Conjunction with ICCV 2005*, vol. 3766 of *Lecture Notes in Computer Science*, (Beijing, China), pp. 1–15, Springer, 2005.

[55] B. Dumas, D. Lalanne, and S. Oviatt, "Multimodal interfaces: A survey of principles, models and frameworks," in *Human Machine Interaction*, pp. 3–26, Springer, 2009.

[56] S. J. Fenves, "A core product model for representing design information," Technical Report NIST IR 6736, National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[57] "ISO/IEC 25063. Systems and software engineering - Systems and software product Quality Requirements and Evaluation (SQuaRE) - Common Industry Format (CIF) for usability: Context of use description," standard, International Organization for Standardization (ISO), Geneva, Switzerland, 2014.

[58] R. D. Moen and C. L. Norman, "Circling back," *Quality Progress*, vol. 43, no. 11, p. 22, 2010.

[59] D. Garvin, "What does product quality really means?," *Sloan management review*, pp. 25–43, 1984.

[60] B. Kitchenham and S. L. Pfleeger, "Software quality: The elusive target [special issues section]," *IEEE software*, vol. 13, no. 1, pp. 12–21, 1996.

[61] W. Shewhart, *Economic Control of Quality of Manufactured Product*. London, UK: Van Nostrand, 1931.

[62] N. Bevan, "Quality in use: Meeting user needs for quality," *Journal of Systems and Software*, vol. 49, no. 1, pp. 89–96, 1999.

[63] V. A. Zeithaml, "Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence," *Journal of Marketing*, vol. 52, no. 3, p. 2, 1988.

[64] "ISO 9000:2015 Quality management systems," standard, International Organization for Standardization (ISO), Geneva, Switzerland, 2015.

[65] "ISO/IEC 9126-1:2001 - Software engineering - product quality - Part 1: Quality Model," standard, International Organization for Standardization (ISO), Geneva, Switzerland, 2001.

[66] "ISO/IEC 25010:2011(E). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models," standard, International Organization for Standardization (ISO), Geneva, Switzerland, 2011.

[67] J. M. González Calleros, J. Guerrero García, and J. Vanderdonckt, "Advance human–machine interface automatic evaluation," *Universal Access in the Information Society*, vol. 12, pp. 387–401, Nov. 2013.

[68] D. Kieras, "Using the keystroke-level model to estimate execution times," *University of Michigan*, vol. 5, 2001.

[69] R. Geng and J. Tian, "Improving Web Navigation Usability by Comparing Actual and Anticipated Usage," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 1, pp. 84–94, 2015.

[70] B. Finch, "Internet discusstions as a source for consumer product customer involvement and quality information: An exploratory study," *Journal of Operations Management*, vol. 17, no. 5, pp. 535–556, 1999.

[71] R. L. Ackoff, "From data to wisdom," *Journal of applied systems analysis*, vol. 16, no. 1, pp. 3–9, 1989.

[72] G. Bellinger, D. Castro, and A. Mills, "Data, information, knowledge, and wisdom." www.systems-thinking.org/dikw/dikw.htm, 2004.

[73] C. Zins, "Conceptual approaches for defining data, information, and knowledge," *Journal of the American Society for Information Science and Technology*, vol. 58, pp. 479–493, Feb. 2007.

[74] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[75] T. D. Wilson, "The nonsense of knowledge management," *Information research*, vol. 8, no. 1, pp. 8–1, 2002.

[76] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in Information Systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[77] C. Ramirez and B. Valdes, *A General Knowledge Representation Model of Concepts*. INTECH Open Access Publisher, 2012.

[78] G. Hoekstra, *Ontology Representation: Design Patterns and Ontologies That Make Sense*. IOS Press, 2009.

[79] M. Masterman, "Semantic message detection for machine translation, using an interlingua," in *Proceedings of the International Conference on Machine Translation*, (Teddington, UK), pp. 438–475, 1961.

[80] F. Lehmann, "Semantic networks in artificial intelligence," *Computers & Mathematics with Applications*, vol. 23, no. 2-5, pp. 1–50, 1992.

[81] M. Minsky, "A framework for representing knowledge," Technical Report AIM-306, Massachusetts institute of technology, Cambridge, USA, 1974.

[82] R. Fikes and T. Kehler, "The role of frame-based representation in reasoning," *Communications of the ACM*, vol. 28, no. 9, pp. 904–920, 1985.

[83] F. Baader, I. Horrocks, and U. Sattler, "Description logics," *Foundations of Artificial Intelligence*, vol. 3, pp. 135–179, 2008.

[84] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data & knowledge engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.

[85] N. F. Noy, D. L. McGuinness, *et al.*, *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, Stanford, CA, 2001.

[86] S. P. Gardner, "Ontologies and semantic data integration," *Drug discovery today*, vol. 10, no. 14, pp. 1001–1007, 2005.

[87] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *ACM SIGMod Record*, vol. 33, no. 4, pp. 58–64, 2004.

[88] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of Protégé: An environment for knowledge-based systems development," *International Journal of Human-computer studies*, vol. 58, no. 1, pp. 89–123, 2003.

[89] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," 2004.

[90] R. Isermann, "Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance," *Springer Science and Business Media*, 2006.

[91] J. S. Bennett and C. R. Hollander, "DART: An Expert System for Computer Fault Diagnosis.," in *Proceedings of the International Joint Conferences on Artificial Intelligence*, (Vancouver, Canada), pp. 843–845, 1981.

[92] B. Freyermuth, "Knowledge based Incipient Fault Diagnosis of Industrial Robots," *IFAC Proceedings Volumes*, vol. 24, no. 6, pp. 369–375, 1991.

[93] H. Monsef, S. Jadid, and A. M. Ranjbar, "Fuzzy rule-based expert system for power system fault diagnosis," *IEE Proceedings - Generation, Transmission and Distribution*, vol. 144, no. 2, pp. 186 – 192, 1997.

[94] D. G. Rajpathak, "An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain," *Computers in Industry*, vol. 64, no. 5, pp. 565–580, 2013.

[95] Basili, Victor R. and Ramsey, Connie Loggia, "ARROWSMITH-P: A prototype expert system for software engineering management," in *Proceedings of the Expert Systems in Government Symposium*, (Washington, DC, USA), pp. 252–264, IEEE Computer Society, 1985.

[96] Kathuriaa, R., Anandarajanb, M., and Igbariac, M., "Selecting IT applications in manufacturing: A KBS approach.," *International Journal of Management Science*, pp. 605–616, 1999.

[97] I. Vlahavas, I. Stamelos, I. Refanidis, and A. Tsoukias, "ESSE: An expert system for software evaluation," *Knowledge-Based Systems*, vol. 12, no. 4, pp. 183–197, 1999.

[98] A. S. Jadhav and R. M. Sonar, "Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach," *Journal of Systems and Software*, vol. 84, no. 8, pp. 1394–1407, 2011.

[99] K. Eldrandaly and S. Naguib, "A knowledge-based system for GIS software selection.," *The International Arab Journal of Information Technology*, vol. 10, no. 2, pp. 152–159, 2013.

[100] S. M. Butt and W. F. W. Ahmad, "Analysis and Evaluation of Cognitive Behavior in Software Interfaces using an Expert System," *International Journal*, vol. 5, no. 1, pp. 146–154, 2012.

[101] W. Leopairote, A. Surarerks, and N. Prompoon, "Evaluating software quality in use using user reviews mining," in *Proceedings*, (Mahasarakham, Thailand), pp. 257–262, IEEE, 2013.

[102] S. Cox, C. Little, F. Pan, and J. Hobbs, "Time Ontology in OWL," Technical Report OGC 16-071r2, The World Wide Web Consortium (W3C), 2017.

[103] A. Perego and M. Lutz, "ISA Programme Location Core Vocabulary," technical report, The World Wide Web Consortium (W3C), 2015.

[104] D. Lüddecke, N. Bergmann, and I. Schaefer, "Ontology-Based Modeling of Context-Aware Systems," in *Model-Driven Engineering Languages and Systems*, vol. 8767 of *Lecture Notes in Computer Science*, (Valencia, Spain), pp. 484–500, Springer, Cham, 2014.

[105] A. Gerlicher, "Entwicklung eines Frameworks zur Analyse der Nutzung von Infotainment-Systemen auf Basis des „Infotainment-Recorders"," Technical report, Steinbeis-Transferzentrum, 2014.

[106] S. Busse, "HMI Analytics Framework - Handbuch," Manual, c4c Engineering Gmbh, Wolfsburg, 2016.

[107] E. Sirin, A. Kalyanpur, B. Parsia, B. C. Grau, and Y. Katz, "Pellet: A Practical OWL-DL Reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, p. 26, 2007.

[108] A. Gross, "Daten. People's HMI Flottenstudie," Technical report, Volkswagen AG, Wolfsburg, 2015.

[109] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[110] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Advances in Database Technology (EDBT '96)* (P. Apers, M. Bouzeghoub, and G. Gardarin, eds.), (Berlin, Heidelberg), pp. 1–17, Springer Berlin Heidelberg, 1996.

[111] T. Deriyenko, *Exploring the Potential of Social Media Analytics in the Context of Product Monitoring.* Master Thesis, Technische Universität Braunschweig, Braunschweig, 2014.

[112] G. Peek, *Text Mining - Analyse produktbezogener Texte aus dem Internet zur automatischen Bestimmung semantischer Cluster.* Bachelor thesis, Hochschule Hannover (University of Applied Sciences and Art), Hannover, 2016.

[113] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," *International journal of human-computer studies*, vol. 43, no. 5-6, pp. 907–928, 1995.